

DESCRIPTION OF THE
BDX 930 PROCESSOR AT
THE GATE LOGIC LEVEL

FINAL REPORT

BY:

F. SWERN
J. MC GOUGH

THE BENDIX CORPORATION
FLIGHT SYSTEMS DIVISION
TETERBORO, N.J. 07608

CONTRACT NO. NAS1-16807

NASA
NATIONAL AERONAUTICS
AND SPACE AGENCY
LANGLEY RESEARCH CENTER
HAMPTON, VIRGINIA 23665

APRIL 1982

(NASA-CR-181642) DESCRIPTION OF THE BDX 930
PROCESSOR AT THE GATE LOGIC LEVEL Final
Report (Bendix Corp.) 325 p

N88-7(994

Unclas
00/60 0145733

TABLE OF CONTENTS

<u>SECTION</u>	<u>TITLE</u>	<u>PAGE</u>
1.0	INTRODUCTION	6
1.1	Background	6
1.2	Objectives	6
2.0	BDX 930 ARCHITECTURE	7
3.0	PROCESSOR BOUNDARIES	12
4.0	EXTERNAL INTERFACES	20
4.1	Clock Inputs	20
4.2	Memory Interface	20
4.3	I/O Interface	20
4.4	Interrupt System	21
4.5	Test Set Control	
4.6	Test Points	27
5.0	EMULATION SYSTEM SYNTAX DEFINITION	29
5.1	Library of Chip Definitions	29
5.2	Circuit Card Definitions	33
5.3	Processor Definition	36
5.4	System Definition	37
6.0	VALIDATION OF THE EMULATION SYSTEM SYNTAX DESCRIPTION OF THE BDX 930 COMPUTER	38
7.0	CHIP LIBRARIES	40
8.0	INTERCONNECTION DESCRIPTION	66
9.0	EXTENSION TO A COMPLETE SIFT PROCESSOR	67
10.0	REFERENCES	69
APPENDIX A	CPU CARD CHIP LIBRARY IN EMULATION SYSTEM SYNTAX	70
APPENDIX B	CPU CARD CHIP LIBRARY IN BLISS	111
APPENDIX C	TIMING AND CONTROL CARD CHIP LIBRARY IN EMULATION SYNTAX AND IN BLISS	176
APPENDIX D	BDX 930 PROCESSOR INTERCARD DESCRIPTION	184
APPENDIX E	CPU CARD DESCRIPTION	192
APPENDIX F	TIMING AND CONTROL CARD DESCRIPTION	243

TABLE OF CONTENTS (Continued)

<u>SECTION</u>	<u>TITLE</u>	<u>PAGE</u>
APPENDIX G	PROM DATA	250
APPENDIX H	CONTENTS OF THE COMPUTER TAPE SUPPLIED FOR SOURCE CODE DISTRIBUTION	271
APPENDIX I	SELF-TEST PROGRAM	274

LIST OF FIGURES

<u>FIGURE</u>	<u>TITLE</u>	<u>PAGE</u>
1	PARALLEL OPERATION OF THE BDX 930 PROCESSOR	9
2	COMPONENTS OF THE BDX 930 CPU	11
3	PROCESSOR ARCHITECTURE	13
4	SCHEMATIC DIAGRAM, CPU CARD	14
5	SCHEMATIC DIAGRAM, TIMING AND CONTROL CARD	19
6	MICROCIRCUITS AND EQUIVALENT GATE COUNT	43
7	SSI AND PROM SCHEMATIC DIAGRAMS	44
8	54LS113	48
9	54LS151, 54S151	49
10	54LS153	50
11	54LS158	51
12	54LS169	52
13	54LS175	53
14	54LS253	54
15	54LS273	55
16	54LS352	56
17	54LS374	57
18	25LS377	58
19	9407 EQUIVALENT CIRCUIT - BDX 930 APPLICATION	59
20	9407 SCHEMATIC DIAGRAM	60
21	2901A OVERVIEW	62
22	2901A SCHEMATIC DIAGRAM	63
23	2901 RAM REPRESENTATION	64
24	AM2902	65
25	SIFT SYSTEM	68
26	SIFT COMPUTER	68

LIST OF TABLES

<u>TABLE</u>	<u>TITLE</u>	<u>PAGE</u>
1	COMPUTER CONNECTOR PINS	22
2	SYNTAX DEFINITION	34
3	INTEGRATED CIRCUIT CHIP LIBRARY	41
4	54-S-00	71
5	54-LS-00	72
6	54-S-02	73
7	54-LS-02	74
8	54-S-04	75
9	54-LS-08	76
10	54-S-32	77
11	54-LS-86	78
12	54-LS-113	79
13	54-125	80
14	54-S-151	81
15	54-LS-151	82
16	54-LS-153	83
17	54-LS-158	84
18	54-LS-169	85
19	54-LS-175	87
20	54-LS-245	88
21	54-LS-253	89
22	54-LS-273	90
23	54-S-288	91
24	54-LS-352	92
25	54-LS-367	93
26	54-LS-374	94
27	25-LS-377	95
28	54-S-472	96
29	54-LS-472	97
30	9407	98
31	AM-2901-A	101
32	AM-2902	110
33	MODULE TCS113	112
34	MODULE TCS151	114
35	MODULE TCS151	116
36	MODULE TXS153	118
37	MODULE TCS158	120
38	MODULE TCS169	122
39	MODULE TC175	126
40	MODULE TC253	126
41	MODULE TC273	130

LIST OF TABLES (Continued)

<u>TABLE</u>	<u>TITLE</u>	<u>PAGE</u>
42	MODULE TC352	133
43	MODULE TC374	135
44	MODULE TC377	138
45	MODULE T9407	141
46	MODULE T2901A	147
47	74-LS-00	177
48	74-LS-08	178
49	74-S-10	179
50	74-LS-11	180
51	74-S-20	181
52	74-S-37	182
53	74-40	183
54	CARD ASSIGNMENT TABLE	186
55	CARD INTERCONNECTION TABLE	187
56	COMPUTER CONNECTOR PINS	188
57	CPU CHIP LABELS	194
58	MODULE P1T	196
59	MODULE P2T	201
60	MODULE P3T	212
61	MODULE P4T	223
62	CPU BOARD	230
63	CPU BOARD CONNECTOR PINS	239
64	TIMING & CONTROL CHIP LABELS	245
65	T&C BOARD	246
66	MODULE P2T	247
67	TIMING & CONTROL BOARD CONNECTOR PINS	249
68	SEQUENCER CONTROL PROM	252
69	START TABLE	253
70	MICROCODE MEMORY	262
71	COVERAGE OF THE MEMORY ADDRESS PROCESSOR OPERATIONS	280
72	COVERAGE OF ALU CONTROL FIELDS	281
73	COVERAGE OF SCRATCHPAD ADDRESSING CONTROL FIELD	282
74	COVERAGE OF CONDITION SELECT FIELD	282
75	COVERAGE OF STATUS FLAG LOGIC FIELD	283
76	COVERAGE OF SHIFT IN MUX FIELD	283
77	ORDER OF INSTRUCTION EXECUTION	284
78	ADDITIONAL MICROMEMORY COVERAGE	285

1.0 INTRODUCTION

1.1 Background

Logic emulation is a useful tool in assessing the reliability of Fault Tolerant Computer Systems. However, using current software emulators, the host computer requires an execution time that is ten thousand or more times that of the emulated processor. For a computer system like SIFT, the resources required to do fault injection experiments in software may be prohibitive in some testing applications. Injecting faults in the SIFT hardware is one way to circumvent the problem; however, without special hardware, faults can only be injected at the chip level, and not at the gate level.

NASA-Langley Research Center is developing and studying a system for emulating and analyzing Fault Tolerant Computer Systems, using a QM-1 Emulator. The QM-1 is a high-speed, general-purpose digital computer that operates with two levels of microprogram control. The second level of microprogram control is normally used for emulating another computer's instruction set in a very efficient manner. The proposed system will use this second level microprogram as a general purpose logic emulator, with a description of the emulated processor residing in the first level microprogram storage. This arrangement would result in an efficient vehicle for studying faults injected into computer systems at the gate level.

1.2 Objectives

A gate-level description of SIFT is required to be resident in the microprogram storage of the QM-1. The emulation includes a syntax for describing the SIFT system and a translator for constructing the microprogram storage from the syntax description.

The development of the emulation system is proceeding in stages, and a description of the full SIFT computer is not immediately needed. This study represents the first step, and is a description of the BDX-930 Processor in the emulation system syntax.

2.0 BDX-930 ARCHITECTURE

The BDX-930 Digital Processor is a microprogrammed, pipelined machine designed around the AMD2901A four bit microprocessor slice. The machine contains sixteen general purpose registers of which four registers may be loaded directly from memory and two registers may be used as base registers. One register is used as a stack pointer.

The program counter and memory address register are contained in the 9407, a chip designed to perform memory address arithmetic. Along with a temporary register contained on the same chip, the BDX-930 is able to perform four basic addressing modes involving three registers and various instruction fields.

The machine contains three memory interface data registers which are used to input and output memory data. There are also a number of one bit status flag registers that can be manipulated under program control. This includes the F1 and F2 registers, which are hardware flags, and the interrupt enable, overflow status registers. There also exist the indirect and link registers used by the microcode for branching.

The microcode is contained in seven proms and a pipeline register is included for simultaneous microcode fetch and decoding. Various internal and external conditions can affect microcode branching as selected by the microcode itself and a microcode control prom. In addition to a rich instruction set which includes 16 and 32 bit fixed point operations, there is a test set interface in the microcode. A selectable saturate mode is available which limits the results of arithmetic operations when overflow or underflow occur.

Instruction execution is accomplished by a pipelined architecture; various stages of execution occur simultaneously for a sequence of instructions. Consider, for instance, four instructions, A,B,C,D, to be executed in sequence. During the same clock cycle it is possible for the program counter to be incremented to point to instruction D, while instruction C is being fetched, instruction B is being decoded and instruction A is being executed.

With this level of parallelism, it will be noted that when the execution phase of an instruction is one clock cycle, the average time to perform the entire instruction will be one clock cycle. This relation can better be understood by referring to Figure 1.

It should also be noted that the BDX-930 is roughly partitioned into the four stages of the pipe: address, fetch, decode, and execute.

Partition 1 - Address Processor

- 4 - 9407 Memory Address Processor Equivalent Circuit
- Selector Chips to Multiplex Memory Address Source
 - 4 - 54LS352 4:1
 - 2 - 54LS158 2:1

Partition 2 - Data and Status Registers

- 2 - 54LS374 Memory Input Buffer Register
- 2 - 54LS374 Memory Output Buffer Register
- 2 - 54LS374 Next Instruction Register
- 3 - 54LS113 Single Bit Registers for
 - overflow
 - indirect addressing
 - link (bit carry for divide)
 - interrupt mode
 - F1 and F2
- 2 - 54LS153 Select Overflow, Link, and Indirect Bit Sources.
- 2 - 54LS245 octal bus transceivers

Partition 3 - Microcontroller

- Pipeline Register
 - 4 - 54LS273 octal latch
 - 4 - 54LS175 quad latch
 - 1 - 54LS374 octal latch with tri-state
- 1 - 54LS273 External Signal Synchronizer
- 3 - 54LS151 Selectors 8:1 for Branch Conditions
- 1 - 54LS169 Counter for Shift and Multiply Instructions
- 1 - 54LS169 Counter for Multiple Register Load-Store Instructions
- 1 - 54LS377 Instruction Register
- 1 - 54LS253 Microcode Branch Selector

Partition 4 - Execute

- 4 - AMD2901A 4 Bit Slice ALU
- 1 - AMD2902 Lookahead Carry
- 2 - 54LS153 Selector 4:1 Register Selectors
- 1 - 54LS253 Selector 4:1 Shift Bit Selector

PARALLEL OPERATION OF THE BDX930 PROCESSOR

BDX - 930 INSTRUCTION FLOW DIAGRAM

MICRO CYCLE				1	2	3	4	5	6	7	8
CALCULATE INSTRUCTION ADDRESS	A	B	C	D			E	F			G
FETCH	Z	A	B	C			D	E			F
DECODE	Y	Z	A	B			C	D			E
EXECUTE	X	Y	Z	A	B		B	C	D	D	D
EFFECTIVE EXECUTION TIME				A	B			C	D		

XYZ - PREVIOUS INSTRUCTIONS
E, F, G - SUBSEQUENT INSTRUCTIONS

- CONCURRENT CALCULATE INSTRUCTION ADDRESS, FETCH, DECODE AND EXECUTE OF INSTRUCTIONS
- 8 MICRO CYCLES - TOTAL EFFECTIVE EXECUTION TIME

FIGURE 1

These stages of the pipe are joined by various buses throughout the CPU. These buses are formed from tri-state logic and some are bidirectional. An enumeration of the major buses includes

- Y - Connects the output of the ALU (AMD2901A) to the address processor and the output register. In addition, it connects the output of the next instruction buffer to the start address register and instruction register.
- D - Connects the memory data register and the program counter to the input of the ALU.
- DAT - Bidirectional bus connecting memory and I/O to the memory data register and output register.
- M - Bidirectional memory data bus
- MAR - Memory Address Bus
- U - Microcode Bus
- IR - Instruction Register

A list of the devices used in the BDX-930 and their failure rates is given in Figure 2. The data was obtained from MIL-HDBK127B, Notice 2.

COMPONENTS OF THE BDX930 CPU

<u>DEVICE</u>	<u>FAILURE RATE/PER UNIT (PPMH)</u>
9407	1.3931
2901A	2.1656
2902	0.3898
5440	0.0653
54125	0.0855
54S00	0.0855
54S04	0.1003
54S10	0.0764
54S20	0.0654
54S32	0.2138
54S288 (32x8 prom)	0.1787
54S472 (512x8 proms)	1.008
54LS00	0.084
54LS02	0.084
54LS04	0.0983
54LS08	
54LS11	0.0752
	0.084
54LS86	0.084
54LS113	0.1447
54LS151	0.1483
54LS153	0.1447
54LS158	0.1410
54LS169	0.6603
54LS175	0.1703
54LS245	0.3792
54LS253	0.1447
	0.1636
54LS273	0.6882
	0.2681
54LS352	0.3117
54LS367	0.1100
54LS374	0.7234
54LS377	0.7148

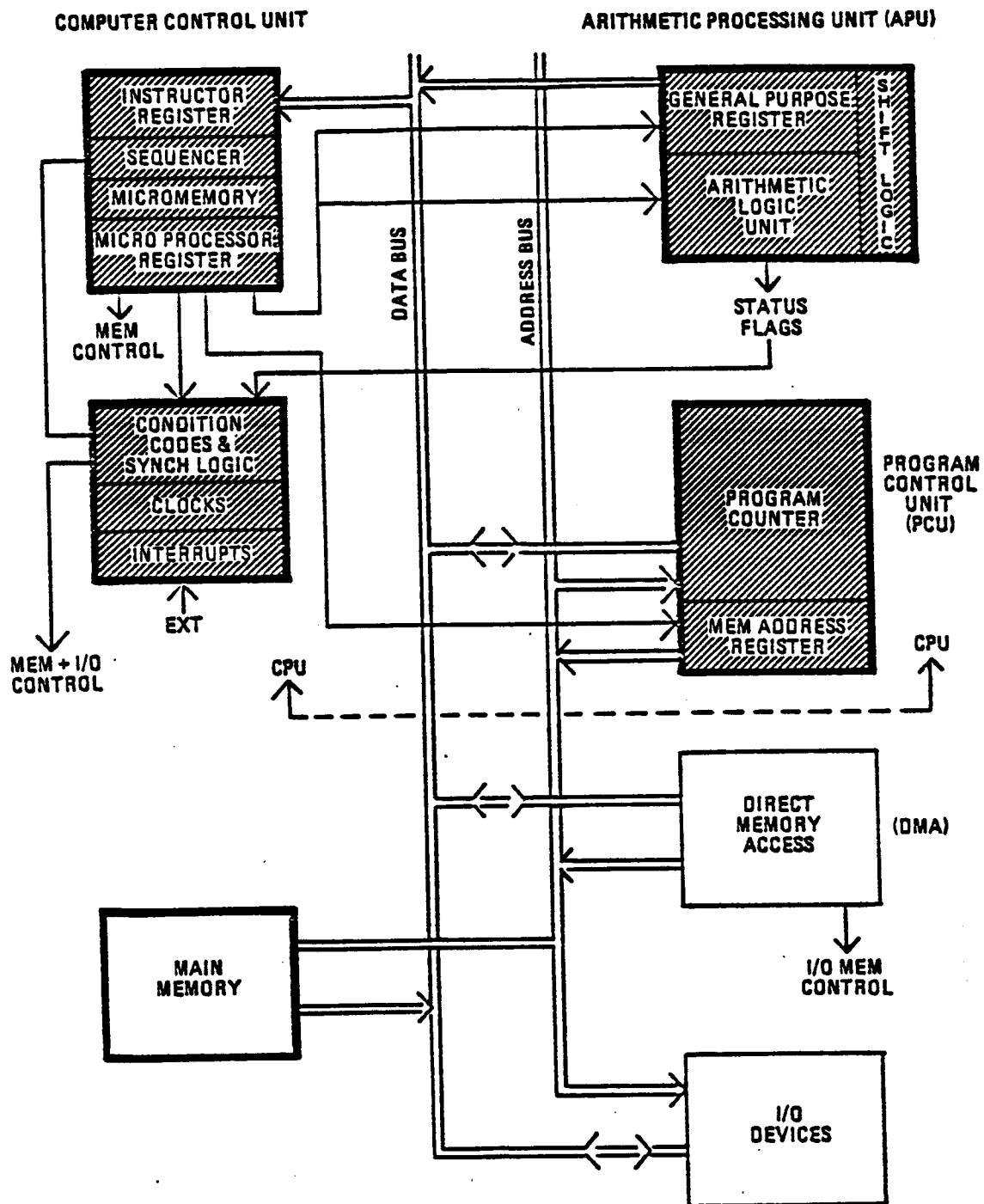
FIGURE 2

3.0 PROCESSOR BOUNDARIES

This description of the BDX-930 Processor includes only the components of the CPU (Central Processor Unit). This includes the arithmetic processing unit, the program control unit and the computer control unit. These areas of the processor, shown in Figure 3, are contained on the CPU and timing/control cards.

Main memory, I/O interface, DMA and data buffering for the main memory and I/O were not simulated. Neither were the interrupt priority logic and the master clock.

In all, logic, equivalent to approximately 5100 gates, was described. These boundaries are equivalent to Bendix's existing gate-level emulation as described in (Ref. 1). Schematic diagrams of the SIFT processor, showing the CPU card, described in its entirety, and the portion of the timing and control card included in this description, are given in Figures 4 and 5.



PROCESSOR ARCHITECTURE

FIGURE 3

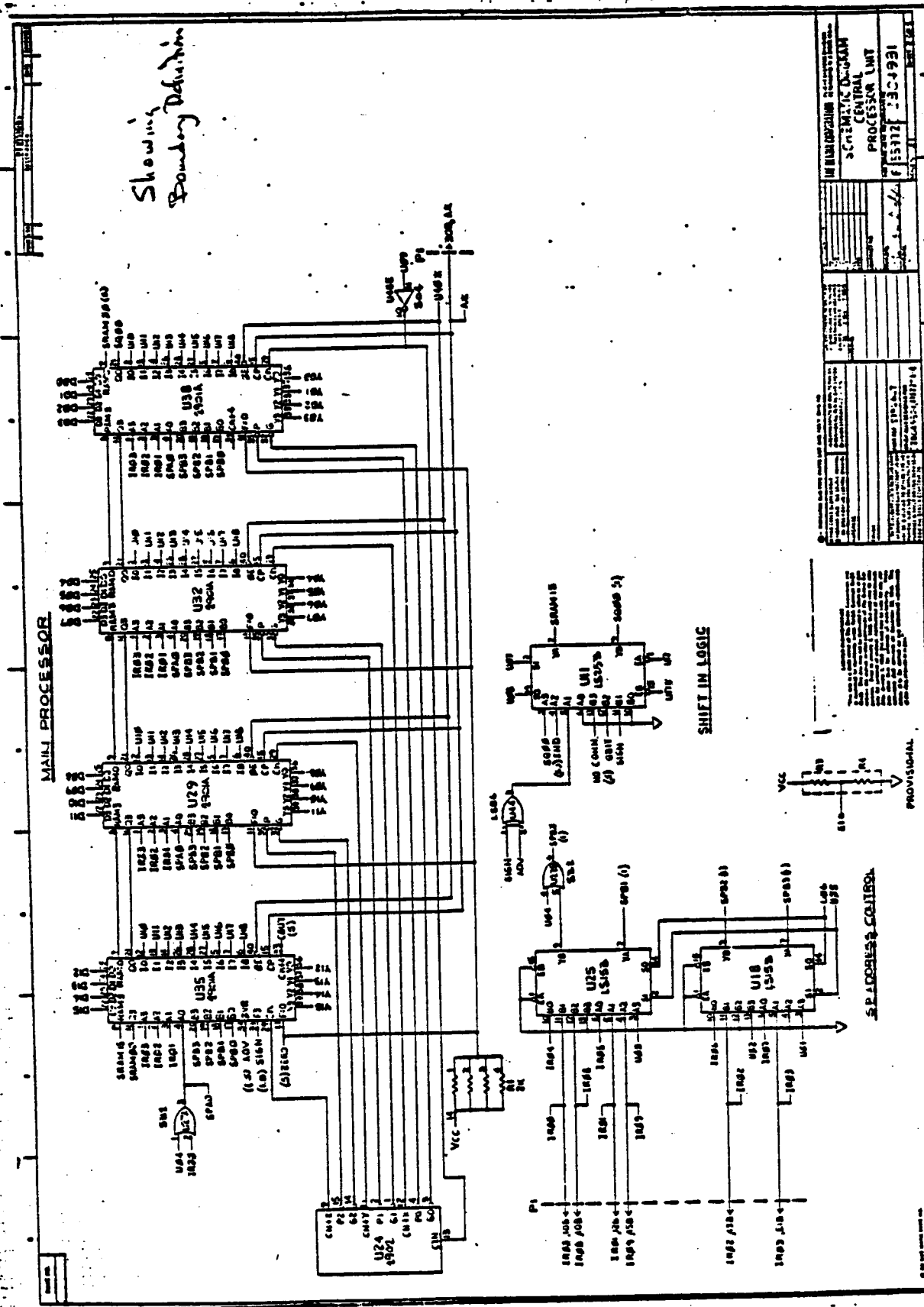


FIGURE 4 (Continued)

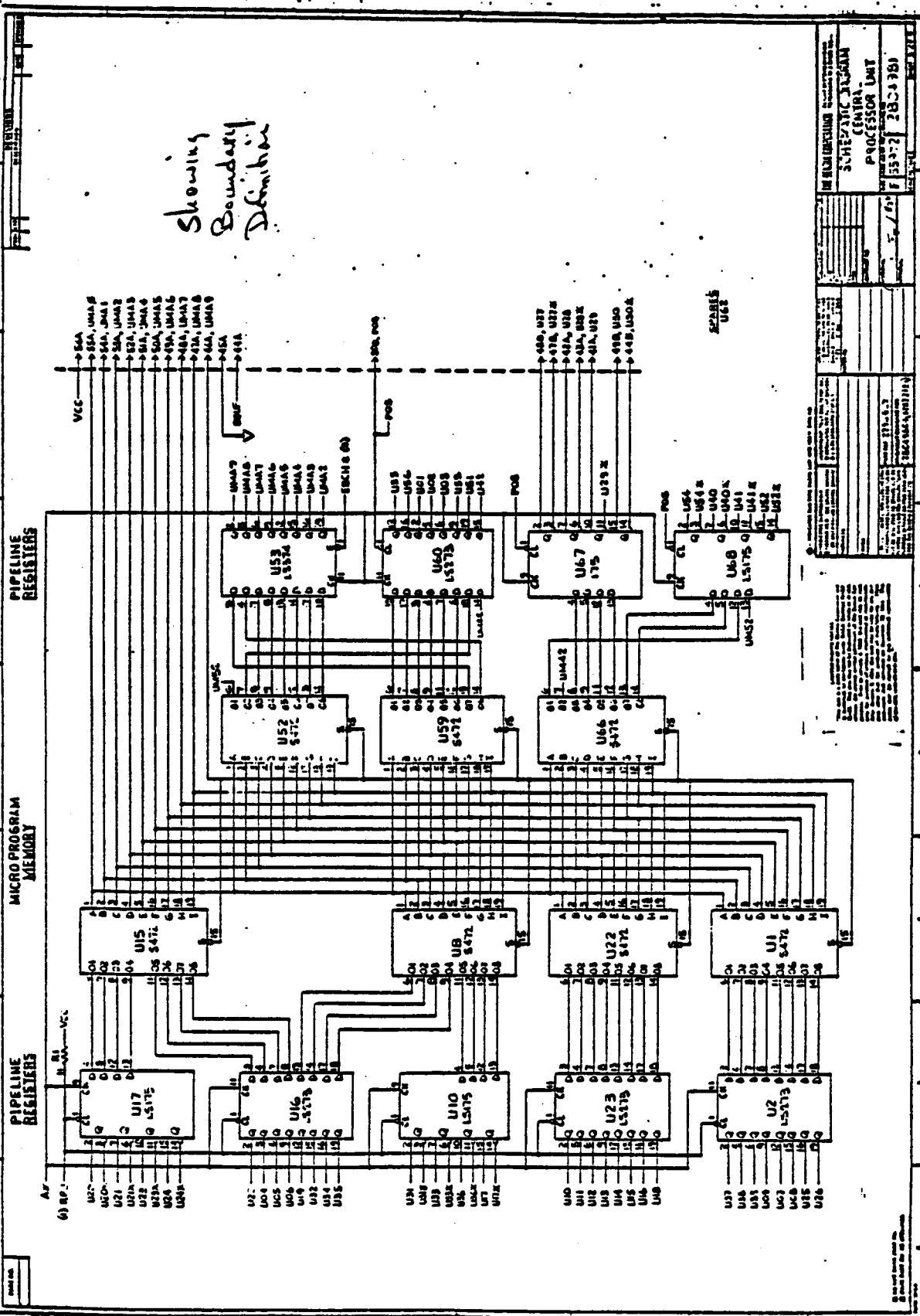


FIGURE 4 (Continued)

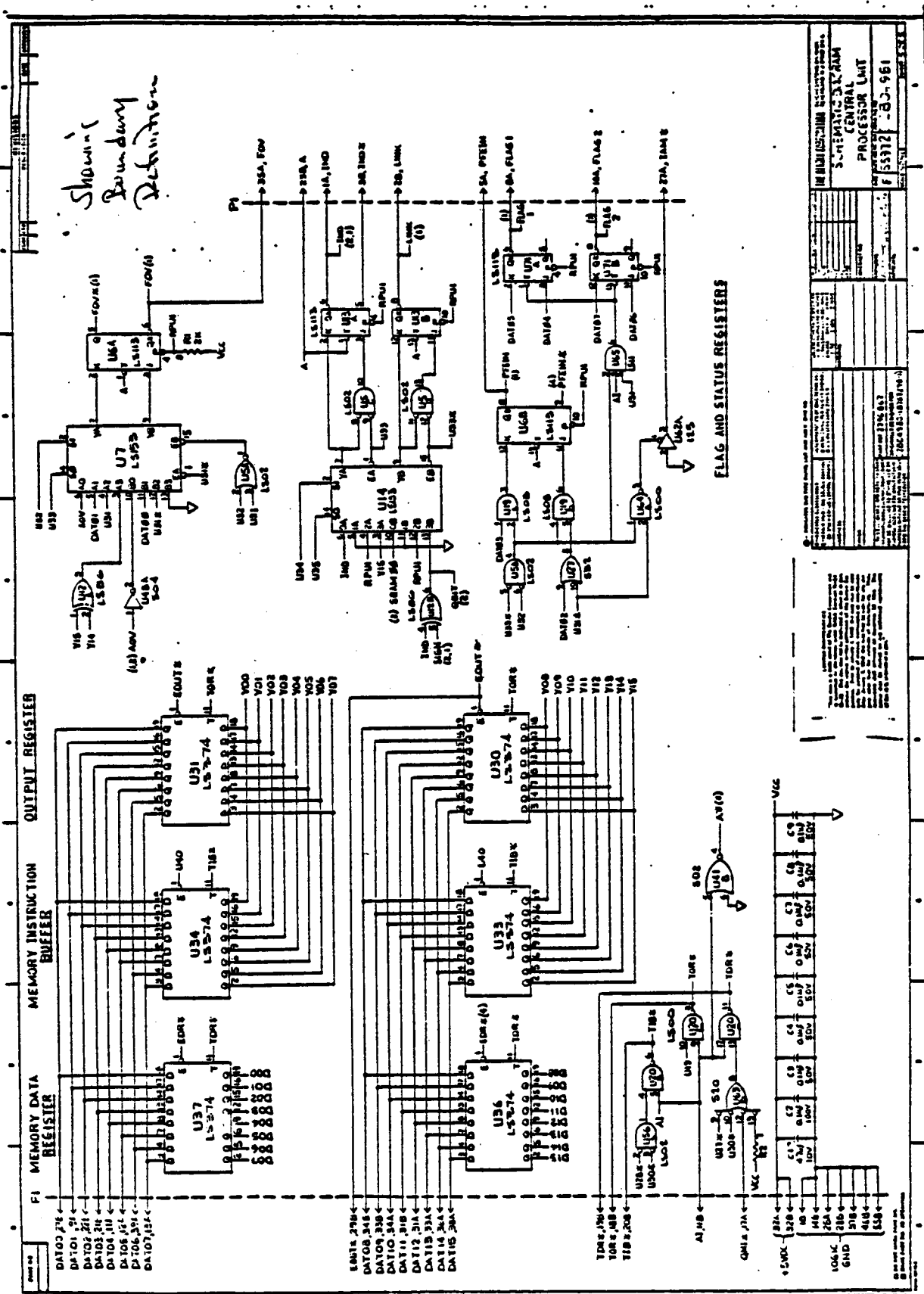
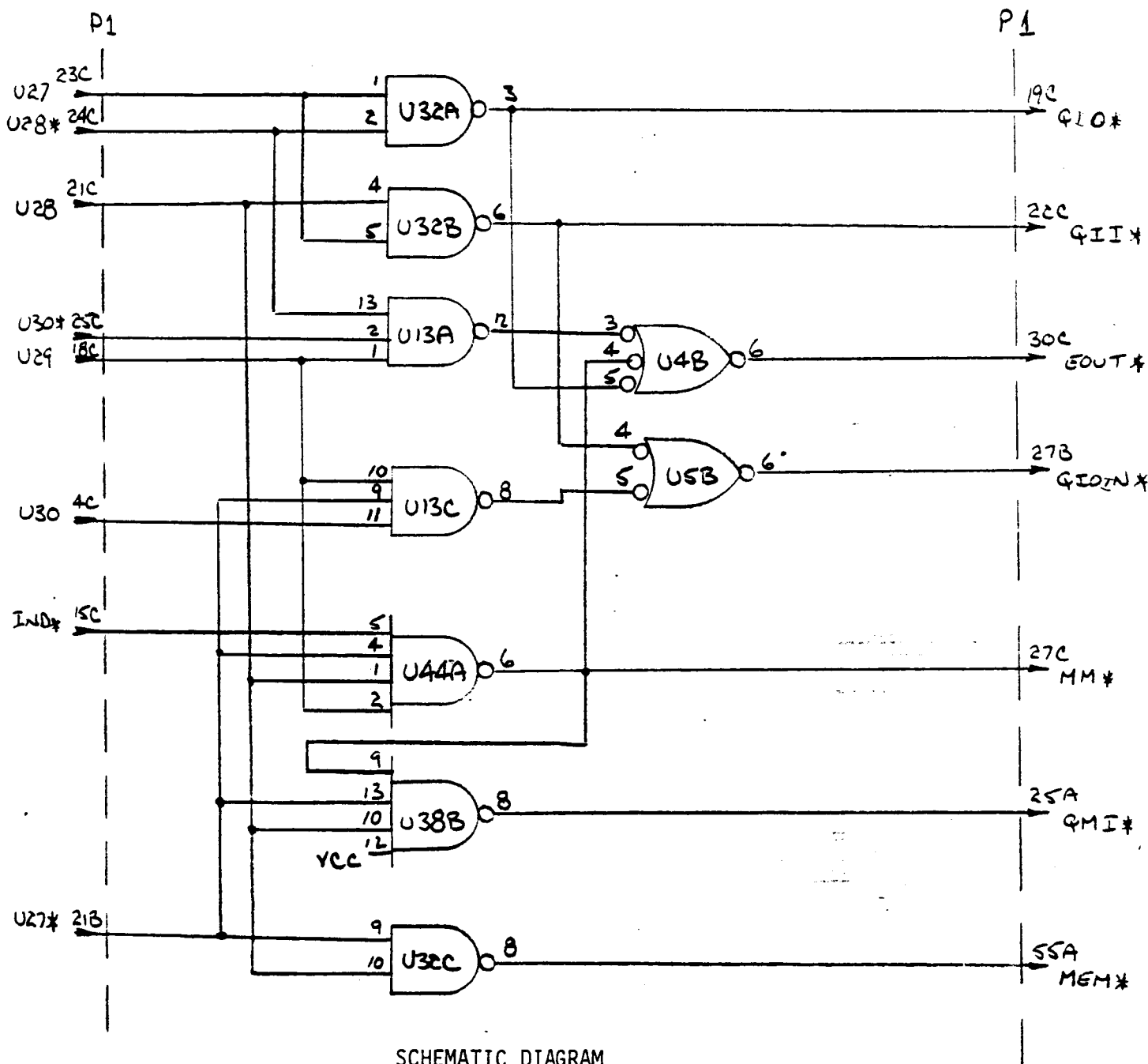


FIGURE 4 (Continued)



SCHEMATIC DIAGRAM
TIMING AND CONTROL CARD

FIGURE 5

4.0 EXTERNAL INTERFACES

Successful execution of a BDX-930 emulation requires that certain external logic signals be supplied. A listing of these signals is given in Table 1, grouped according to function.

4.1 Clock Inputs

The oscillator and divider chain (resident on the timing and control card) was not included in this description. A simulated four megahertz signal must enter J10 pins 11B (AI) and 23B (A). The "power-on sequence" is followed after a signal with a high to low sequence is input on J10 pin 50B (POS).

4.2 Memory Interface

Sixteen signals are presented by the CPU with a memory address inverted on connector J10 pins 4B, 5B, 6B, 7B, 13B, 15B, 16B, 17B, 35B, 36B, 38B, 39B, 52B, 53B, 54B and 56B (MAR00* - MAR15*). Memory data enters and exits the processor on a bi-directional I/O bus contained on connector J10 pins 9A, 11A, 12A, 13A, 21B, 22B, 27B, 31A, 31B, 33A, 33B, 34A, 34B, 36A, 38A and 39A (DAT00-DAT 15).

A memory request is indicated by a low signal on connector J9 pin 55A (MEM*). When a write operation is being requested, a low signal is presented on connector J9 pin C7C (MM*); otherwise, a read operation is being requested. The modified clocking scheme used in this model assumes that the memory operation is complete at the end of a single clock cycle, i.e. the handshake logic is not included in the description.

4.3 I/O Interface

Only a rudimentary I/O interface was included in the description. Sixteen bit I/O data enters and exits the processor on the DAT bus (DAT00-DAT15), as in the memory interface. For output, a low signal is generated on connector J9 pin 19C (QIO*). A low signal is generated for an input instruction on connector J9 pin 22C (QII*). A low signal is presented whenever the I/O bus is to be read on connector J9 Pin 27B (QIOIN*).

The remaining I/O strobes (necessary for the proper execution of the I/O instructions) were not simulated.

Three discrete external signals can be individually tested using skip instructions and enter the CPU on connector J10 pins 2A, 3A and 4A (EXT1, EXT2, EXT3). A discrete output is available on connector J10 pin 10A (FLAG 2). A status input is available for use with certain I/O instructions (ISR, OSR) on connector J10 pin 25B (READY).

4.4 Interrupt System

The interrupt system can be armed and disarmed externally via a logic signal on connector J10 pin 24A (ARM). A logical "OR" of all interrupt requests enters the CPU by an inverted logic signal on connector J10 pin 29A (IR*). When processing the interrupt, a signal is provided to determine the page of the interrupt address (by tying it to the I/O bus in the back plane) on connector J10 pin 27A (IAM*).

4.5 Test Set Control

These signals interface a test set or software development system and can be used to simulate such a system.

4.6 Test Points

These signals are available for monitoring various points in the CPU.

COMPUTER CONNECTOR PINS⁽¹⁾

DATA BUS CONNECTIONS

CONN PIN	SIG IDENT	IC PIN	FUNCTION
J10-7B	MAR00*	U43 - 20	OUTPUT
J10-6B	MAR01*	U43 - 18	OUTPUT
J10-5B	MAR02*	U43 - 16	OUTPUT
J10-4B	MAR03*	U43 - 14	OUTPUT
J10-17B	MAR04*	U42 - 20	OUTPUT
J10-16B	MAR05*	U42 - 18	OUTPUT
J10-15B	MAR06*	U42 - 16	OUTPUT
J10-13B	MAR07*	U42 - 14	OUTPUT
J10-39B	MAR08*	U40 - 20	OUTPUT
J10-38B	MAR09*	U40 - 18	OUTPUT
J10-36B	MAR10*	U40 - 16	OUTPUT
J10-35B	MAR11*	U40 - 14	OUTPUT
J10-56B	MAR12*	U39 - 20	OUTPUT
J10-54B	MAR13*	U39 - 18	OUTPUT
J10-53B	MAR14*	U39 - 16	OUTPUT
J10-52B	MAR15*	U39 - 14	OUTPUT

(1) See Figures 4 and 5 for Signal Identities.

TABLE 1

J10-27B	DAT00	U37 - 18	I/O	17
J10-9A	DAT01	U37 - 17	I/O	18
J10-22B	DAT02	U37 - 14	I/O	19
J10-21B	DAT03	U37 - 13	I/O	20
J10-11A	DAT04	U37 - 8	I/O	21
J10-12A	DAT05	U37 - 7	I/O	22
J10-39A	DAT06	U37 - 4	I/O	23
J10-13A	DAT07	U37 - 3	I/O	24
J10-34B	DAT08	U36 - 18	I/O	25
J10-33B	DAT09	U36 - 17	I/O	26
J10-34A	DAT10	U36 - 14	I/O	27
J10-31B	DAT11	U36 - 13	I/O	28
J10-31A	DAT12	U36 - 8	I/O	29
J10-33A	DAT13	U36 - 7	I/O	30
J10-36A	DAT14	U36 - 4	I/O	31
J10-38A	DAT15	U36 - 3	I/O	32

TABLE 1 (Continued)

I/O AND MEMORY CONTROL SIGNALS

CONN PIN	SIG IDENT	IC PIN	FUNCTION
J9-19C	OIO*	U04 - 5	OUTPUT 3 B
J9-22C	OII*	U32 - 6	OUTPUT
J9-27B	OIOIN*	U05 - 6	OUTPUT
J9-27C	MM*	U38 - 9	OUTPUT
J9-55A	MEM*	U32 - 8	OUTPUT
J10-2A	EXT1	U72 - 4	INPUT
J10-3A	EXT2	U72 - 3	INPUT
J10-4A	EXT3	U72 - 2	INPUT
J10-10A	FLAG2	U71 - 8	OUTPUT
J10-25B	READY	U09 - 17	INPUT
J10-29A	IR*	U48 - 3	INPUT 1 B
J10-24A	ARM	U63 - 1	INPUT 2 B
J10-27A	IAM*	U48 - 6	OUTPUT 4 C

TABLE 1 (Continued)

TEST SET CONTROL SIGNALS

CONN PIN	SIG IDENT	IC PIN	FUNCTION
J10-19A	TEST*	U48 - 13	INPUT
J10-26B	TSSR	U09 - 18	INPUT
J10-6A	HLTP*	U65 - 12	OUTPUT
J10-25A	HALT*	U48 - 9	OUTPUT

TABLE 1 (Continued)

CLOCK SIGNALS

CONN PIN	SIG IDENT	IC PIN	FUNCTION
J10-11B	AI	U20 - 5	INPUT
J10-23B	A	U13 - 1	INPUT
J10-50B	POS	U60 - 1	INPUT

TABLE 1 (Continued)

TEST POINT SIGNALS

CONN PIN	SIG IDENT	IC PIN	FUNCTION
J10-9B	ZERO	U09 - 7	OUTPUT
J10-10B	IR00	U25 - 11	OUTPUT
J10-12B	IR01	U25 - 5	OUTPUT
J10-14A	I3	U19 - 8	OUTPUT
J10-15A	ETIR*	U46 - 1	OUTPUT
J10-16A	I2	U56 - 4	OUTPUT
J10-18A	COND	U49 - 3	OUTPUT
J10-18B	TOR*	U20 - 8	OUTPUT
J10-19B	TDR*	U20 - 11	OUTPUT
J10-20A	ELSB*	U70 - 4	OUTPUT
J10-20B	TIR*	U20 - 6	OUTPUT
J10-21A	ESPC*	U70 - 3	OUTPUT
J10-22A	ESTRT*	U70 - 2	OUTPUT
J10-23A	EBCH*	U70 - 1	OUTPUT
J10-26A	EDR	U48 - 60	OUTPUT
J10-30A	AGV	U09 - 4	OUTPUT
J10-35A	FQV	U06 - 6	OUTPUT
J10-37A	RPTOV*	U04 - 14	OUTPUT
J10-40A	I1	U64 - 11	OUTPUT
J10-40B	IR08	U25 - 12	OUTPUT
J10-41B	SIGN	U09 - 6	OUTPUT
J10-42B	COUT	U09 - 13	OUTPUT
J10-43B	IR02	U18 - 11	OUTPUT
J10-45B	IR09	U25 - 4	OUTPUT
J10-46A	UMA9	U15 - 15	OUTPUT

TABLE 1 (Continued)

J10-47A	UMA8	U15 - 19	OUTPUT
J10-48A	UMA7	U15 - 18	OUTPUT
J10-49A	UMA6	U15 - 17	OUTPUT
J10-50A	UMA5	U15 - 16	OUTPUT
J10-51A	UMA4	U15 - 5	OUTPUT
J10-51B	TR03	U18 - 5	OUTPUT
J10-52A	UMA3	U15 - 4	OUTPUT
J10-53A	UMA2	U15 - 3	OUTPUT
J10-54A	UMA1	U15 - 2	OUTPUT
J10-55A	UMA0	U15 - 1	OUTPUT
J10-1A	IND	U13 - 6	OUTPUT
J10-2B	LINK	U13 - 8	OUTPUT
J10-5A	PFEIN	U06 - 8	OUTPUT
J10-30B	A*	U38 - 15	OUTPUT
J10-7A	EX*	U19 - 11	OUTPUT

TABLE 1 (Continued)

5.0 EMULATION SYSTEM SYNTAX DEFINITION

The pre-processor for the QM-1 emulation system accepts a description of a computer system separated into several sections. These sections are hierarchical in nature so that the emulation can grow in a modular fashion.

The definition of a computer system consists of the following sets of information:

1. Interconnections of processor units, and
2. Description of the function and operation of each external connection.

The definition of a processor unit consists of the following sets of information:

1. Table of card labels and their respective position designation or name in the card file on the motherboard,
2. Card file or motherboard interconnections, and
3. Description of function and operation of each external connection.

The definition of a circuit card consists of the following sets of information:

1. Library of chip definitions by type,
2. Table of chip labels vs. chip types,
3. Table of list of connections between chips and also card connectors,
4. Card label or identifier, and
5. Description of function and operation of each of the external connections.

5.1 Library of Chip Definitions

The integrated circuit library is a collection of integrated circuit type definitions. These definitions contain information about the I.C., including internal logic and pin assignments. The proper syntax is given by:

Type: Alphanumeric string terminated by an end-of-line.
First character is alphabetic; no spaces allowed
(20 characters max.).

Family: TTL

Power: List of ID's equated to pin numbers, i.e.
VCC = P16, GND = P8.
The list is terminated by end of line that is not
preceded by a comma.

Description: Alphanumeric string terminated.

Unused Pins: Either a list of pins or none, i.e. P1, P2 ...

Functions: A list of assignment statements for each device
modeled on the chip.

The beginning of a chip definition is indicated by \$Define Chip\$.

The end is indicated by \$End Chip\$.

Assignment statement fields conform to the following syntax:

A) Gates

G-label (pin) = Gate type (input list)
 where G-label is 10 characters max.

G: Identifies the device as a GATE.

Label: Identifier, using the characters A..Z, 0..9, '.

(Pin): If the device output is connected to a pin or pins,
then these pins are listed here. Otherwise, this
field is omitted.

Gate Type: One of the following:

AND
OR
NOT
NAND
NOR
XOR
XNDR

(Input List): Inputs to the device are indicated by a list of labels separated by commas. If an input is inverted (i.e. the logic drawing has a circle at the input) then the form is "INV(label)". Otherwise, the form is "label".

This label can be the output of a device on the chip, or a chip pin number. The \bar{Q} output of a flip-flop is indicated by a ' '.

B) Three-State Gates

GTS-label (pins) = gate type (input list); disconnect (input label)
where GTS-label is 10 characters max.

GTS: Identifies the output as a 3-state device.

Label: Same as gates.

(Pins): Same as gates.

Gate Type: Same as gates.

(Input List): Same as gates.

Disconnect: DIS-HIGH if disconnect occurs when "input label" is high
DIS-LOW if disconnect occurs when "input label" is low.

(Input Label): Input signal that enables the 3-state device output.

C) Flip-Flops

FF-Label (Q pins(s); \bar{Q} pins(s)) = Flip-flop type (input list)
where FF-label 10 characters max.

FF: Identifies device as a flip-flop.

Label: Identifies using the characters A..Z, 0..9.

Q pins: Pin number(s) of Q outputs

\overline{Q} pins: Pin number(s) of \overline{Q} outputs

Flip-Flop Type: One of the following: D
R-S
J-K
T

Input List: Each list item is of the form X = input identifies,
where X can be the following:

PRESET: For preset
CLEAR: For clear
J: j input
K: k input
CLK: Clock input
DATA: Data
RESET: Reset input
SET: Set input

List items are separated by commas. Only those
input on the modeled device need to be specified.
The clock input identifier has a modifier of
- DOWN for a down-going clock and -UP for an
up-going clock if the clock is not level.

D. Read-Only Memory Chips

ROM-label (pin) = Address (MSB, Next MSB, ... LSB),
where ROM-label is 10 characters max.,

- OR -

ROM TS-label (pin) = Address (MSB, Next MSB, ... LSB); Operation
(input list),
where ROM TS-label is 10 characters max.,

ROM or ROMTS signify a read-only memory output with normal output or
3-state output respectively.

Pin indicates the chip pin that the output drives (if any).

Address indicates that the output is a function of the value of the
input address vector.

MSB, Next MSB, ..., LSB: The address source identifiers are listed in
the order of most significant bit (MSB) to least significant bit (LSB).

Operation (Input List): If the ROM output is 3-state, this field will indicate that the control variable source and the manner in which the 3-state control is applied.

The data is stored in the ROM will be represented by a Table in the emulation. The Address (list) will tell the Translator what variables are used to form the address for the ROM.

The syntax for chip definitions is summarized in Table 2.

5.2 Circuit Card Definitions

The description of a circuit card contains a number of tables. One table will indicate the interconnections between components and connectors on the card. The entries in this table are the pins of the components and connectors (e.g. U2-19 → U6-10, U3-6 → P1-42, where the first example indicates a connection of U2 and U64 and the second indicates the connection of U3 and connector P1). Another table lists the correspondence between component identifiers for the circuit board and component type (e.g. U10 - 54LS10).

A) Circuit Board Interconnection Table Syntax

The interconnection of the integrated circuits on the circuit card as well as the card connectors(s) are listed in this table. A table entry consists of three fields. The first field is a source or output of a device. The second field is a list of destinations. The third field, which is optional, is the signal name and is inside quote marks.

Source → destination, ..., destination "signal name".

B) Chip Label Table Syntax

The table of integrated circuit identifiers and types has entries of the form

chip id - chip type

where chip id is an identifier that will uniquely label each chip on the card (e.g. U1, U3, Z4, etc.). Chip type is a label which identifies the type of chip and is the type field in the integrated circuit library.

C) Circuit Card Connector Pin Functions

A description of the electrical function for each connector pin on each circuit card is required so that these functions may be modeled

SYNTAX DEFINITION

```

<IC_LIBRARY> := <IC_LIST> <EOF>
<IC_LIST> := <IC_LIST> <DEFINITION>
<IC_LIST> := <DEFINITION>
<DEFINITION> := <HEAD> <BODY> <TAIL>
<HEAD> := $ CHIP DEFINITION $
<BODY> := <CHIP_TYPE> <FAMILY> <POWER> <DESCRIPTION>
+          <UNUSED_PINS> <FUNCTIONS>
<CHIP_TYPE> := TYPE : <CHIP_ID>
<CHIP_ID> := <IDENTIFIER>
<FAMILY> := FAMILY : <IDENTIFIER>
<POWER> := POWER : <POWER_LIST>
<POWER_LIST> := <POWER_LIST> , <POWER_ENTRY>
<POWER_LIST> := <POWER_ENTRY>
<POWER_ENTRY> := VCC = <PIN_NUMBER>
<POWER_ENTRY> := GND = <PIN_NUMBER>
<PIN_NUMBER> := P - <CONSTANT>
<DESCRIPTION> := DESCRIPTION : <WORDS> <END_WORDS>
<WORDS> := <WORDS> <WORD>
<WORDS> := <WORD>
<WORD> := <IDENTIFIER>
<WORD> := <CONSTANT>
<END_WORDS> := .
<UNUSED_PINS> := UNUSED PINS : <PIN_LIST>
<UNUSED_PINS> := UNUSED PINS : NONE
<PIN_LIST> := <PIN_LIST> , <PIN_NUMBER>
<PIN_LIST> := <PIN_NUMBER>
<FUNCTIONS> := FUNCTIONS : <STATEMENT_LIST>
<STATEMENT_LIST> := <STATEMENT_LIST> <ELEMENTARY_STATEMENT>
<STATEMENT_LIST> := <ELEMENTARY_STATEMENT>
<ELEMENTARY_STATEMENT> := <GATE_STATEMENT>
<ELEMENTARY_STATEMENT> := <TS_GATE_STATEMENT>
<ELEMENTARY_STATEMENT> := <FF_STATEMENT>
<ELEMENTARY_STATEMENT> := <ROM_STATEMENT>
<ELEMENTARY_STATEMENT> := <TS_ROM_STATEMENT>
<GATE_STATEMENT> := <GATE_VAR> <ASSIGN_OP> <GATE_TYPE> <ARGUMENTS>
<GATE_VAR> := G - <LABEL> <OUTPUT_PIN>
<GATE_VAR> := G - <LABEL>
<LABEL> := <IDENTIFIER>
<LABEL> := <IDENTIFIER> /
<OUTPUT_PIN> := ( <PIN_NUMBER> )
<ASSIGN_OP> := =
<GATE_TYPE> := AND
<GATE_TYPE> := OR
<GATE_TYPE> := NAND
<GATE_TYPE> := NOT
<GATE_TYPE> := NOR
<GATE_TYPE> := XOR
<GATE_TYPE> := XNOR

```

"+" indicates continuation of previous line.

ATTACHMENT 1, Page 1 of 2

TABLE 2

```

<ARGUMENTS> := ( <INPUT_LIST> )
<INPUT_LIST> := <INPUT_LIST> , <INPUT>
<INPUT_LIST> := <INPUT>
<INPUT> := <SOURCE>
<INPUT> := <INVERT> <SOURCE>
<INVERT> := INV.
<SOURCE> := <PIN_NUMBER>
<SOURCE> := G - <LABEL>
<SOURCE> := GTS - <LABEL>
<SOURCE> := FF - <LABEL>
<SOURCE> := ROM - <LABEL>
<SOURCE> := ROMTS - <LABEL>
<TS_GATE_STATEMENT> := <TS_GATE_VAR> <ASSIGN_OP> <GATE_TYPE>
+
; <CONTROL> <CONTROL_LIST>
<TS_GATE_VAR> := GTS <LABEL> <OUTPUT_PIN>
<CONTROL> := <DISCON_HIGH>
<CONTROL> := <DISCON_LOW>
<DISCON_HIGH> := DIS_HIGH
<DISCON_LOW> := DIS_LOW
<CONTROL_LIST> := ( <INPUT> )
<FF_STATEMENT> := <FF_VAR> <ASSIGN_OP> <FF_TYPE> <FF_ARGUMENTS>
<FF_VAR> := FF <LABEL> <FF_OUTPUT_PINS>
<FF_OUTPUT_PINS> := ( <Q_PIN> ; <NOT_Q_PIN> )
<Q_PIN> := <PIN_NUMBER>
<Q_PIN> :=
<NOT_Q_PIN> := <PIN_NUMBER>
<NOT_Q_PIN> :=
<FF_TYPE> := D
<FF_TYPE> := RS
<FF_TYPE> := JK
<FF_TYPE> := T
<FF_ARGUMENTS> := ( <FF_ARGUMENTS> <FF_ARG> )
<FF_ARGUMENTS> := ( <FF_ARG> )
<FF_ARG> := <FF_PAR> <ASSIGN_OP> <INPUT>
<FF_PAR> := P
<FF_PAR> := C
<FF_PAR> := J
<FF_PAR> := K
<FF_PAR> := DATA
<FF_PAR> := R
<FF_PAR> := SET
<FF_PAR> := CLK.UP
<FF_PAR> := CLK.DOWN
<FF_PAR> := CLK
<ROM_STATEMENT> := <ROM_VAR> <ASSIGN_OP> <ADDRESS_VECTOR>
<ROM_VAR> := ROM <LABEL> <OUTPUT_PIN>
<ADDRESS_VECTOR> := ( <ADDRESS_LIST> )
<ADDRESS_LIST> := <ADDRESS_LIST> , <ADDRESS>
<ADDRESS_LIST> := <ADDRESS>
<ADDRESS> := <INPUT>
<TS_ROM_STATEMENT> := <TS_ROM_VAR> <ASSIGN_OP>
+
<ADDRESS_VECTOR> ; <CONTROL_PART>
<TS_ROM_VAR> := ROMTS <LABEL> <OUTPUT_PIN>
<CONTROL_PART> := <CONTROL> <CONTROL_LIST>
<TAIL> := $ END CHIP $

```

TABLE 2 (Cont.)

during the emulation of the circuit card. Information such as the source of the function, its relationship to the circuit card, etc. are the kinds of information needed.

D) Card Label

Each type card will have an alpha-numeric identifier assigned of up to 10 characters.

5.3 Processor Definition

A) Card Assignment Table Syntax

This table assigns a card type (card label) to a card slot or position.

Example:	<u>Position</u>	<u>Card Type</u>
	A1	CPU
	A2	Memory
	A3	Timing

B) Card Interconnection Table Syntax

The table indicates the signal connections between cards. It is of the following form:

<u>Source</u>		<u>Feeds</u>
A1 - A	→	A2 - A, A3 - A, ...
A1 - B	→	A3 - B
Etc.		

The identifiers are of the form "card-position identifies - pin designation".

This table will also include the connection of the cards to external connectors, if any.

C) External Connection Description

Each external signal needs to be described in sufficient detail to allow simulation of these signals.

5.4 System Definition

A) Processor Interconnections Syntax

The connections between processors will be similar to the card interconnections.

B) External Signal Description Syntax

Similar to external connection description.

6.0 VALIDATION OF THE EMULATION SYSTEM SYNTAX DESCRIPTION OF THE BDX930 COMPUTER

The description of a computer as complex as the BDX930 requires some type of validation procedure to verify its correctness. Using a hierarchical approach, it is possible to show that the description is complete and functionally correct. The chip libraries and interconnection tables were translated into the BLISS Programming Language, and the resulting BDX930 simulation was used to successfully execute a self test program.

One possible source of error in the current study is the lack of a functioning preprocessor for the emulation system. Without such a preprocessor, it is impossible to check for syntax errors in the description.

The BLISS Language was designed by Digital Equipment Corporation for building system software. It is ideally suited for simulating logic networks because:

- o It is very close to assembly language and generates code efficient in execution time,
- o It contains all the necessary primitive logic operators,
- o It is a structured language with a good variety of constructs to make programming easier and more legible,
- o It contains a macro facility so that operators may be defined in a syntax similar to that of the emulator preprocessor.

A set of BLISS routines were coded representing the two cards in the processor. The interchip (and intercard) connection data, as coded in the emulator syntax, were interspersed with the BLISS statements as BLISS comments. After each BLISS chip simulation, the output connections from that chip (and certain input connections) are listed in emulator syntax. For convenience, the cards were broken down into four BLISS partitions, similar to the four stages of the instruction pipeline. After each BLISS partition was validated all signals entering and leaving each BLISS chip were visually checked to be correctly contained in the interchip connection portion of the emulation syntax description.

For simple chips (small scale integration), a single BLISS statement was written for each gate, and included in the proper BLISS partition routine. However, when the chip contained MSI or LSI, a BLISS subroutine was constructed containing a model of the chip. Again emulator syntax statements were interspersed with BLISS statements, corresponding gates being grouped together. Once the BLISS subroutine was validated, the equivalence between statements in the two languages was visually checked.

The BLISS partition routines and chip subroutines were run as an emulation using executive routines coded by Bendix for its gate level parallel emulator. This simplified the checkout procedure. A self test program, developed by Bendix for in-flight testing of its BDX930 minicomputer to a 92% coverage level of hardware faults, was used as a validation tool. Appendix I lists the self test program. The BLISS routines were considered validated when this program was successfully executed.

7.0 CHIP LIBRARIES

Table 3 lists the integrated circuits in the chip library for both the CPU card and the timing and control card. Table 4 gives an equivalent gate count for MSI and LSI chips. Figures 7 through 24 are diagrams of the gate level equivalent for each chip.

- o Appendix A contains descriptions of chips in the preprocessor syntax for the CPU card.
- o Appendix B contains BLISS routines for the chips in Appendix A.
- o Appendix C contains descriptions of chips in the preprocessor syntax for the timing and control card.

TABLE 3

INTEGRATED CIRCUIT CHIP LIBRARY

TYPE ----	DESCRIPTION -----
54-LS-00	QUAD 2-INPUT POSITIVE-NAND GATES.
54-S-00	QUAD 2-INPUT POSITIVE-NAND GATES.
74-LS-00	QUAD 2-INPUT POSITIVE-NAND GATES.
54-LS-02	QUAD 2-INPUT POSITIVE-NOR GATES.
54-S-02	QUAD 2-INPUT POSITIVE-NOR GATES.
54-S-04	HEX INVERTERS.
54-LS-08	QUAD 2-INPUT POSITIVE-AND GATES.
74-LS-08	QUAD 2-INPUT POSITIVE-AND GATES.
74-S-10	TRIPLE 3-INPUT POSITIVE NAND GATES.
74-LS-11	TRIPLE 3-INPUT POSITIVE AND GATES.
74-S-20	DUAL 4-INPUT POSITIVE NAND GATES.
54-S-32	QUAD 2-INPUT POSITIVE-OR GATES.
74-S-37	QUAD 2-INPUT POSITIVE-NAND BUFFERS.
74-40	DUAL 4-INPUT POSITIVE-NAND BUFFERS.
54-LS-86	QUAD 2-INPUT EXCLUSIVE-OR GATES.
54-LS-113	DUAL GATED J - K FLIP-FLOPS.
54-125	QUAD BUS BUFFER GATES WITH THREE-STATE OUTPUTS.

TABIE 3 (CONT'D)

54-LS-151	1 - OF - 8 DATA SELECTORS/ MULTIPLEXERS.
54-S-151	1 - OF - 8 DATA SELECTORS/ MULTIPLEXERS.
54-LS-153	DUAL 1 - OF - 4 DATA SELECTORS/ MULTIPLEXERS.
54-LS-158	2-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS WITH INVERTED OUTPUTS.
54-LS-169	SYNCHRONOUS 4-BIT UP/DOWN COUNTER.
54-LS-175	QUAD D-TYPE FLIP-FLOPS.
74-LS-245	OCTAL BUFFERS, LINE DRIVERS, LINE RECEIVERS WITH INVERTED 3-STATE OUTPUTS.
54-LS-253	DUAL 1-OF-4 DATA SELECTORS/MULTIPLEXERS WITH 3-STATE OUTPUTS.
54-LS-273	OCTAL D-TYPE FLIP-FLOPS
54-S-288	32 WORDS BY 8 BIT PROM WITH 3-STATE OUTPUTS.
54-LS-352	DUAL 4-LINE-TO-1-LINE DATA SELECTORS/MULTIPLEXERS.
54-LS-367	HEX BUS DRIVERS WITH 3-STATE OUTPUTS.
54-LS-374	OCTAL D-TYPE TRANSPARENT LATCHES, AND EDGE-TRIGGERED FLIP-FLOPS WITH 3-STATE OUTPUTS.
25-LS-377	OCTAL D-TYPE FLIP-FLOPS.
54-LS-472	512 WORDS BY 8 BIT PROM WITH 3-STATE OUTPUTS.
54-S-472	512 WORDS BY 8 BIT PROM WITH 3-STATE OUTPUTS.
AM-2901-A	BIPOLAR MICROCONTROLLER.
AM-2902	LOOK-AHEAD CARRY GENERATORS.
9407	MEMORY ADDRESS GENERATORS.

MICROCIRCUITS AND EQUIVALENT GATE COUNT

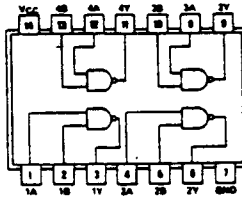
<u>DEVICE</u>	<u>EQUIVALENT GATES</u>
2901A	798
2902	19
54113	8
54151	17
54153	16
54158	15
54169	58
54175	22
54245	18
54253	16
54273	34
54352	16
54374	26
54377	35
9407	143

FIGURE 6

**QUADRUPLE 2-INPUT
POSITIVE-NAND GATES**

00

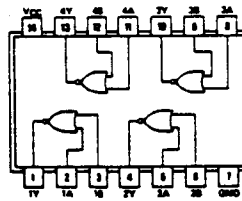
positive logic:
 $Y = \overline{AB}$



**QUADRUPLE 2-INPUT
POSITIVE-NOR GATES**

02

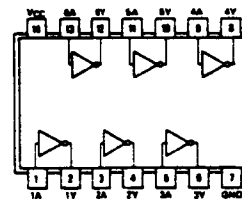
positive logic:
 $Y = \overline{A+B}$



HEX INVERTERS

04

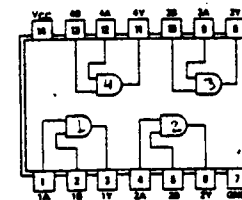
positive logic:
 $Y = \overline{A}$



**QUADRUPLE 2-INPUT
POSITIVE-AND GATES**

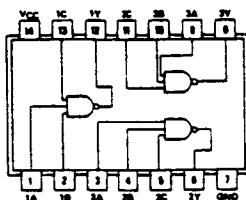
08

positive logic:
 $Y = AB$

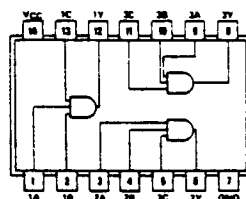


SSI AND PROM SCHEMATIC DIAGRAMS
FIGURE 7

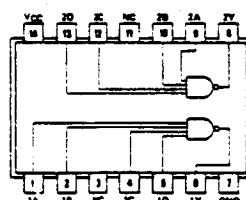
positive logic:
 $Y = \overline{ABC}$



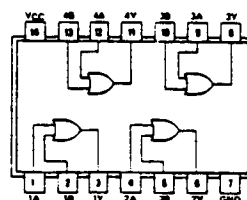
positive logic:
 $Y = ABC$



positive logic:
 $Y = \overline{ABCD}$



positive logic:
 $Y = A + B$

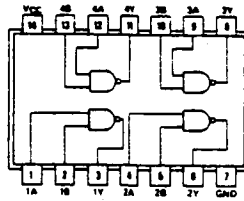


45

**QUADRUPLE 2-INPUT
POSITIVE-NAND BUFFERS**

37

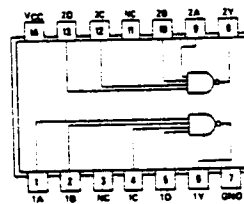
positive logic:
 $Y = \overline{AB}$



**DUAL 4-INPUT
POSITIVE-NAND BUFFERS**

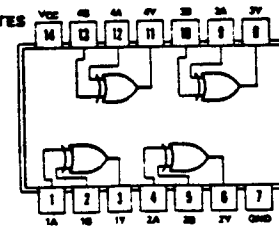
40

positive logic:
 $Y = \overline{ABCD}$



QUADRUPLE 2-INPUT EXCLUSIVE-OR GATES

86 $Y = A \oplus B = \overline{A}B + A\overline{B}$



QUADRUPLE BUS BUFFER GATES

125

positive logic:
 $Y = A$
Output is off (disabled) when C is high.

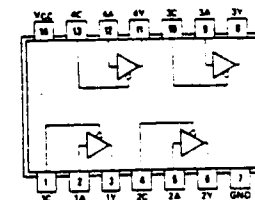
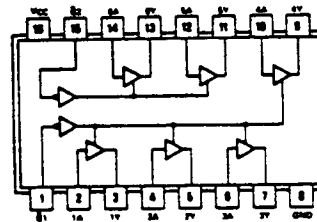


FIGURE 7 (CONT'D)

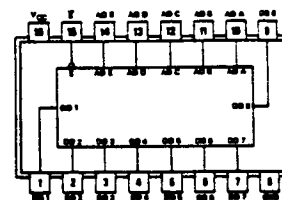
HEX BUS DRIVERS

367 NONINVERTED DATA OUTPUTS
4-LINE AND 2-LINE ENABLE INPUTS
3-STATE OUTPUTS



256-BIT PROGRAMMABLE READ-ONLY MEMORIES

288 32 8-BIT WORDS
3-STATE OUTPUTS



PROGRAMMABLE READ-ONLY MEMORIES

472 3-STATE OUTPUTS

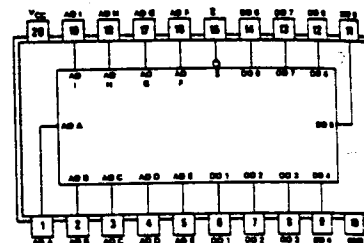
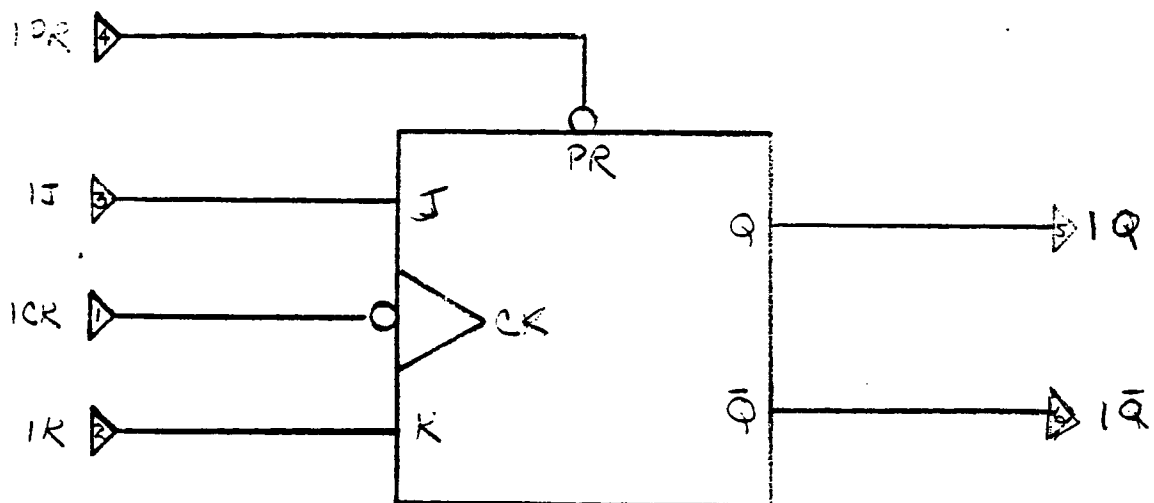
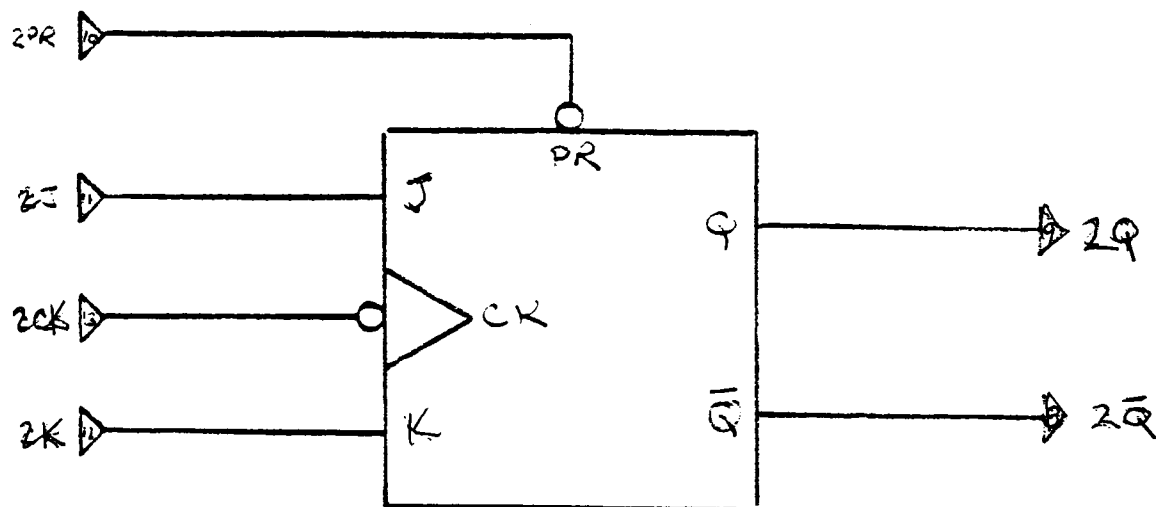
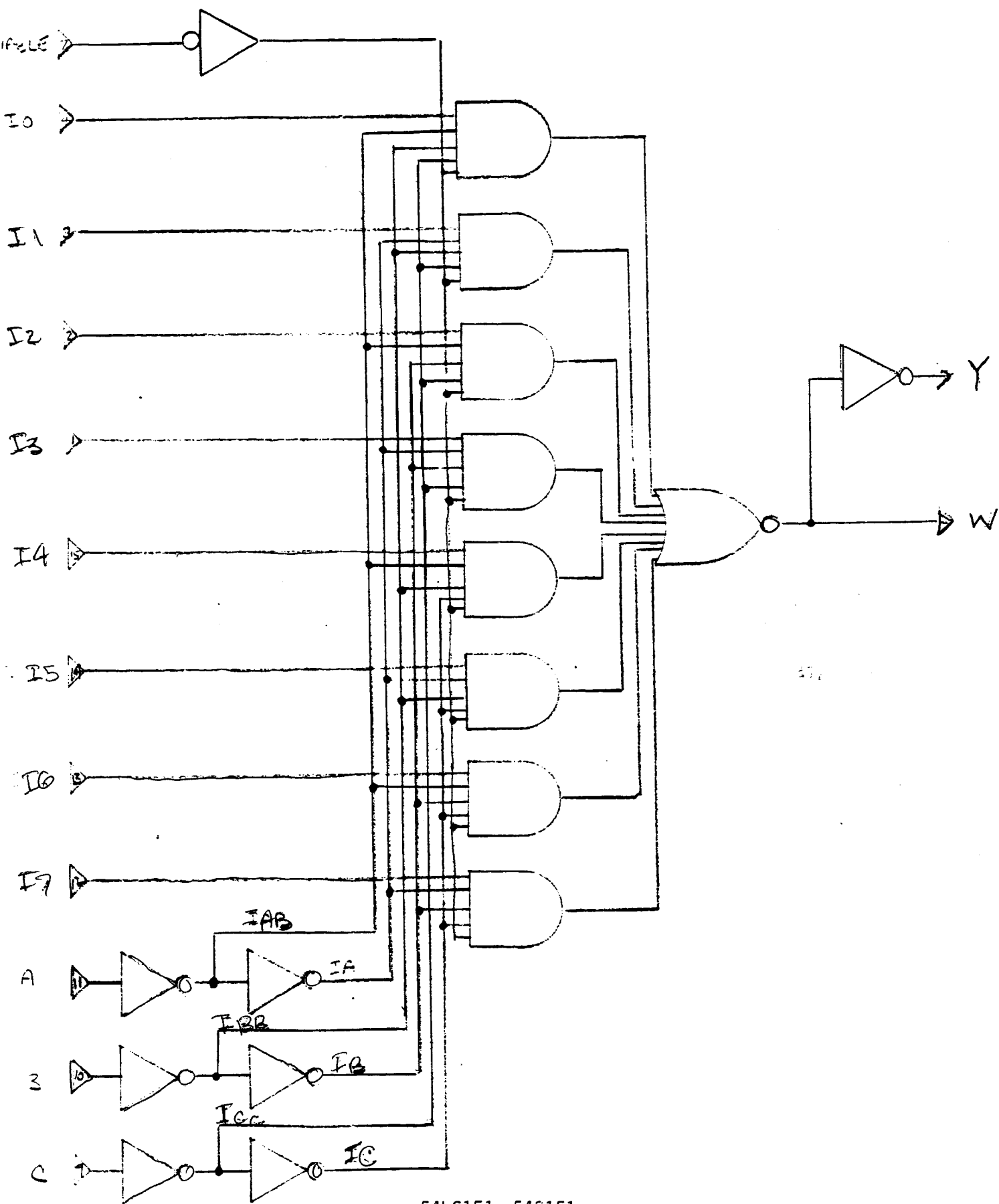


FIGURE 7 (CONT'D)

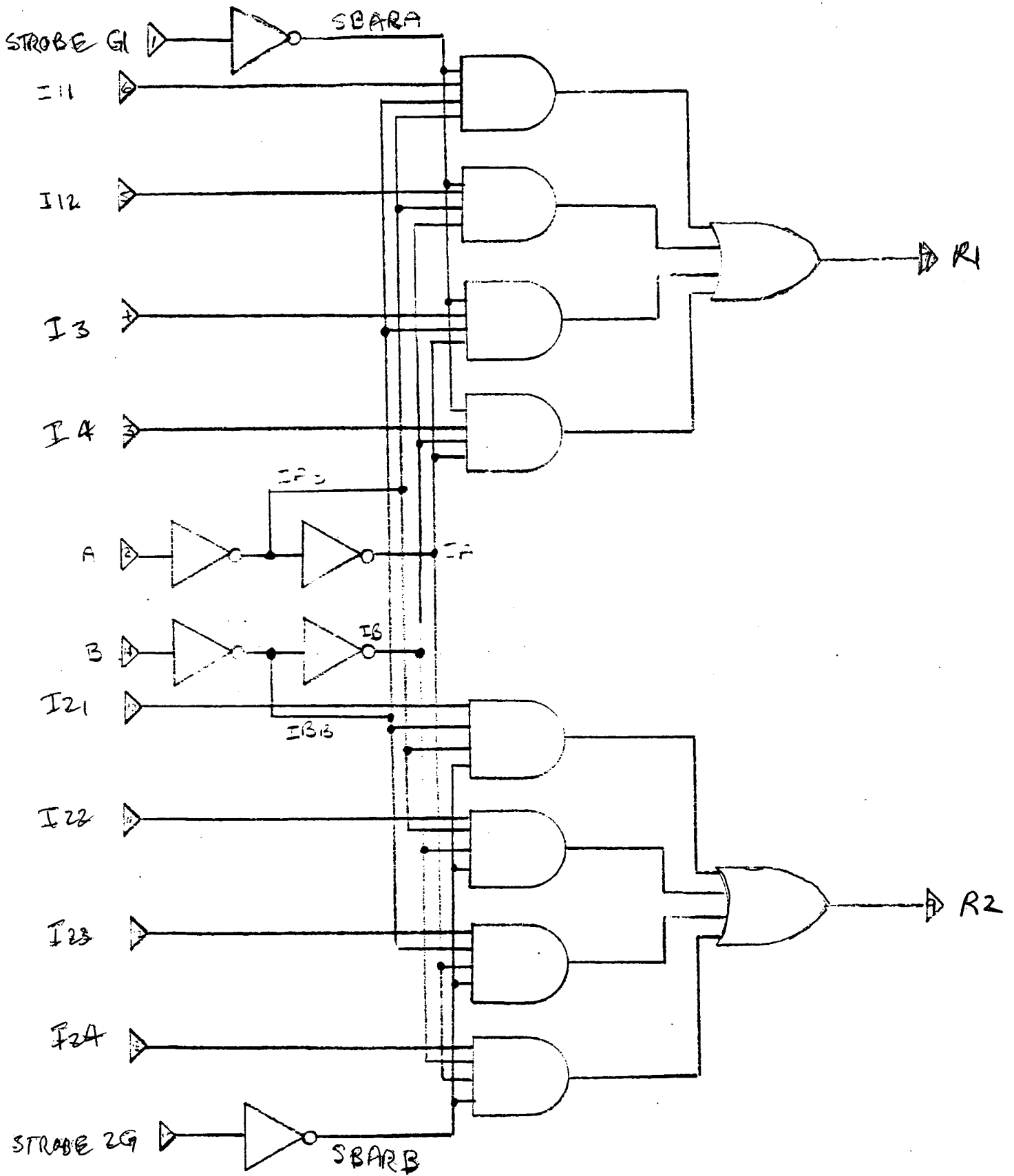


54LS113
FIGURE 8

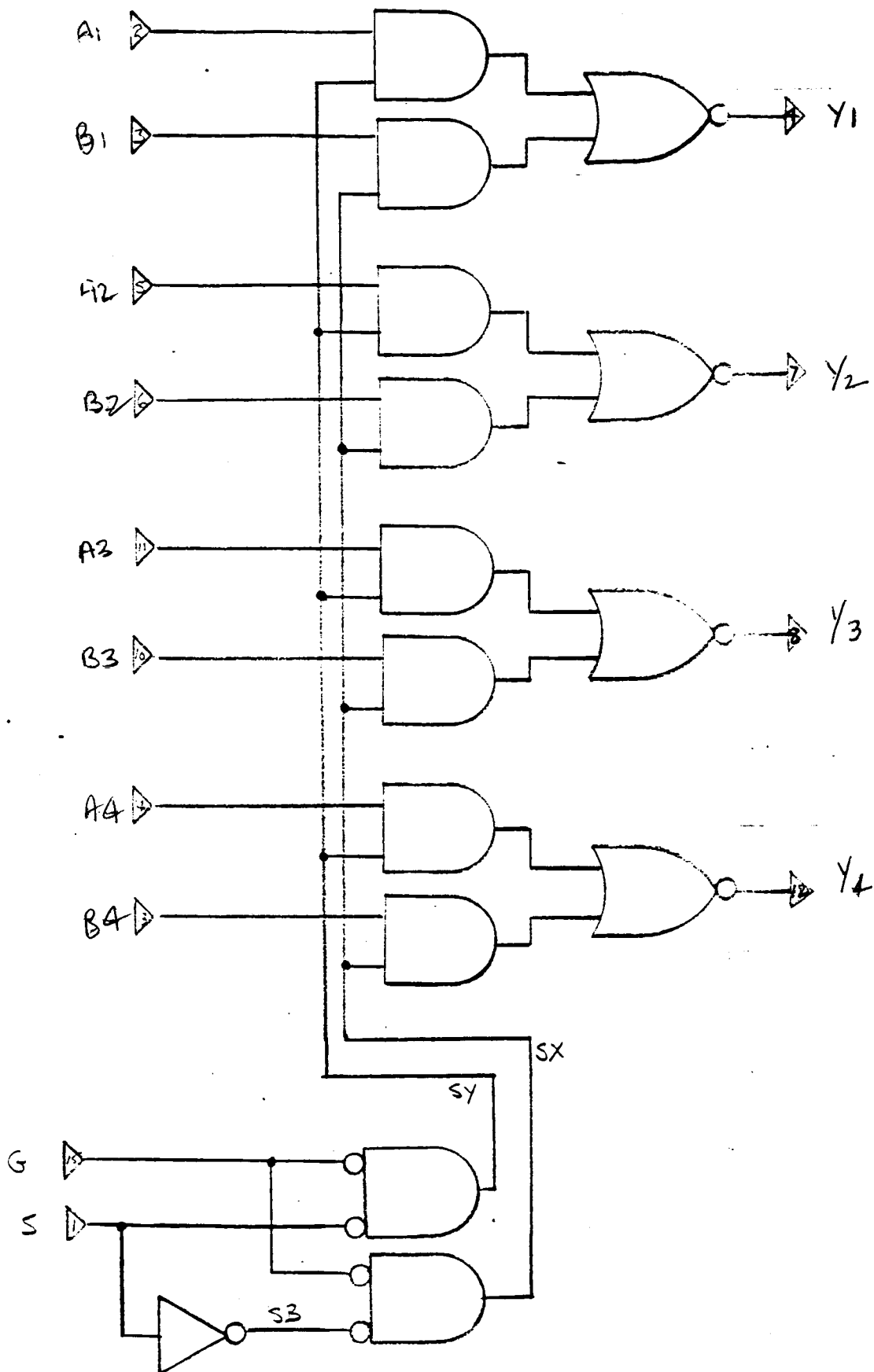


54LS151, 54S151

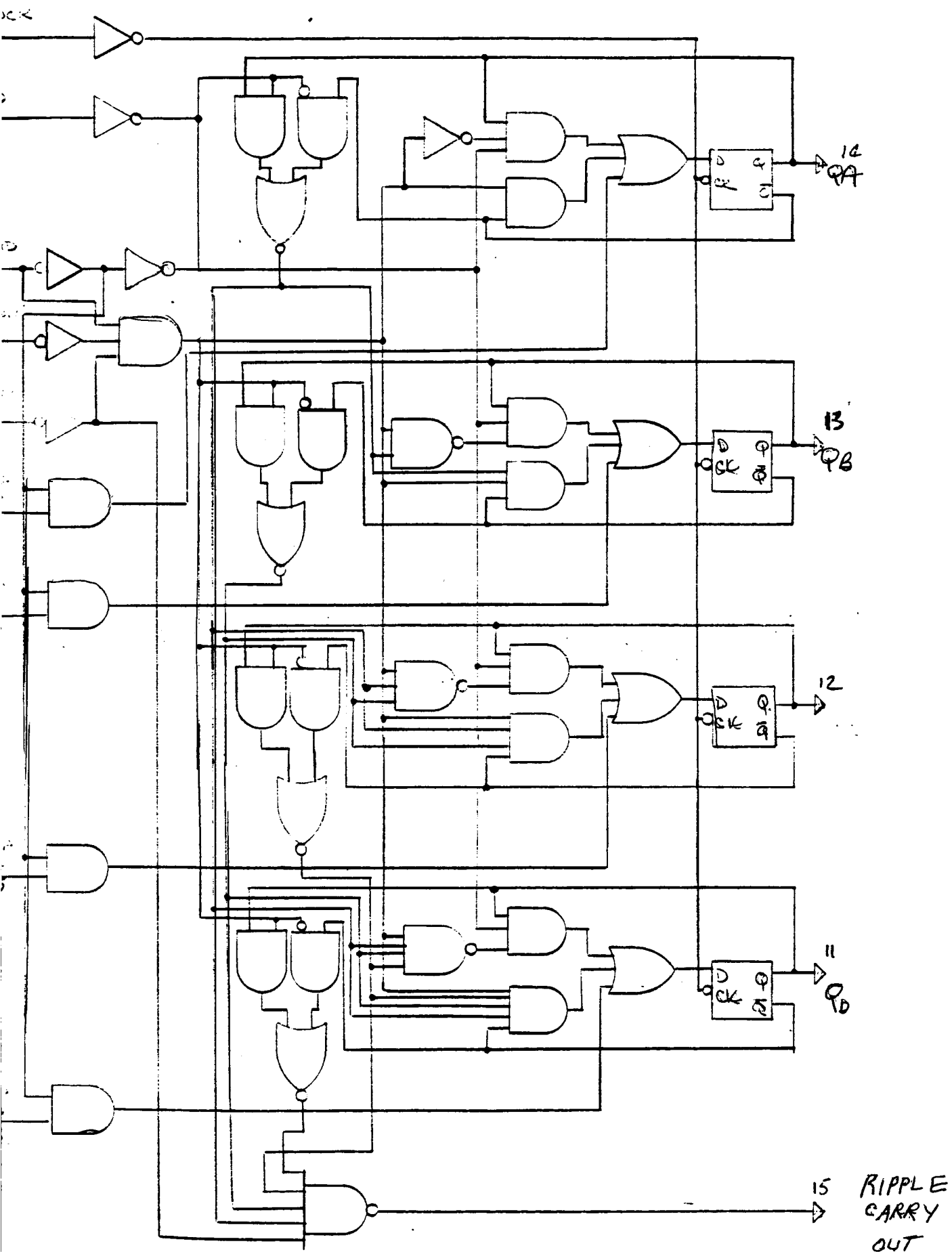
FIGURE 9



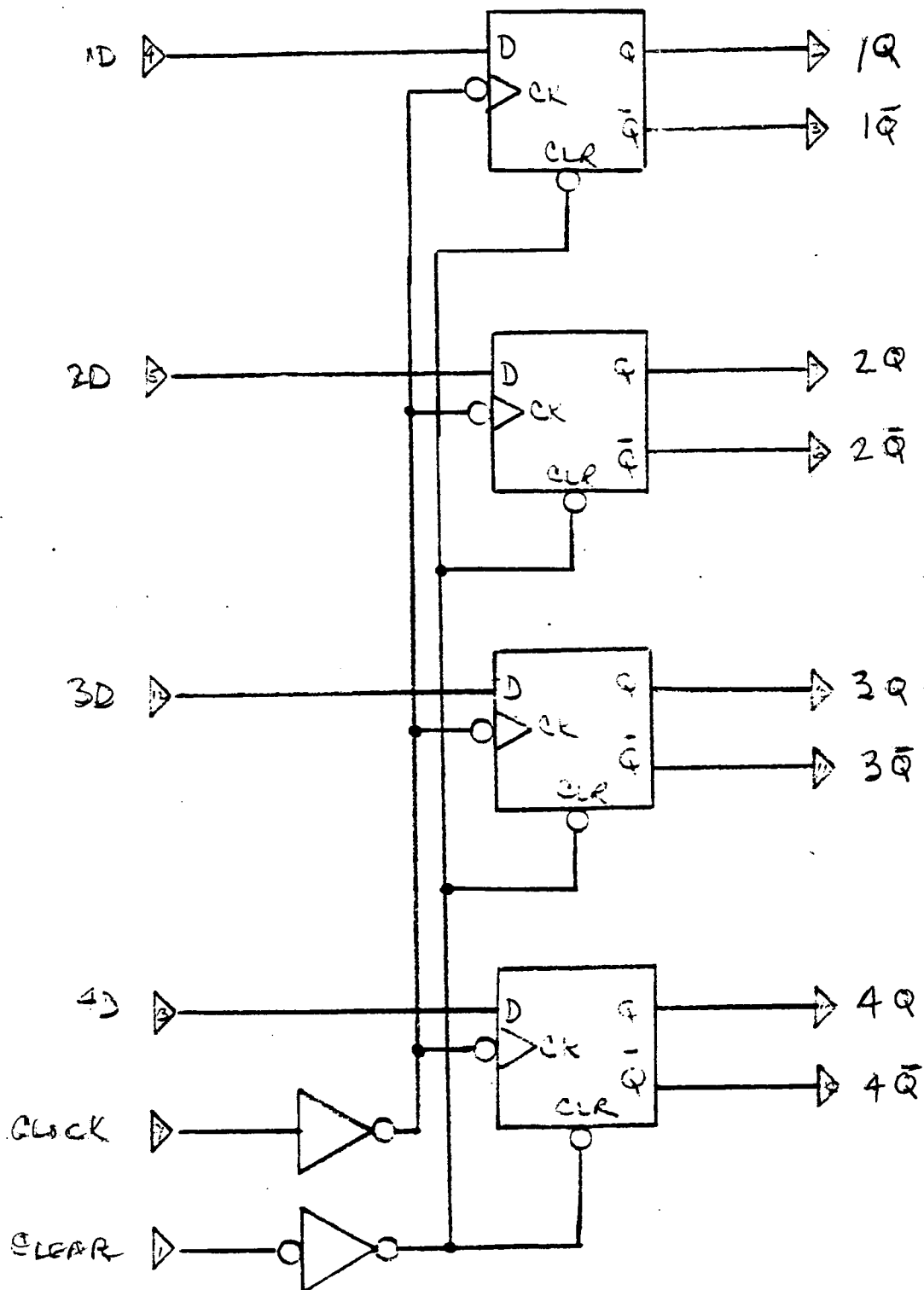
54LS153
FIGURE 10



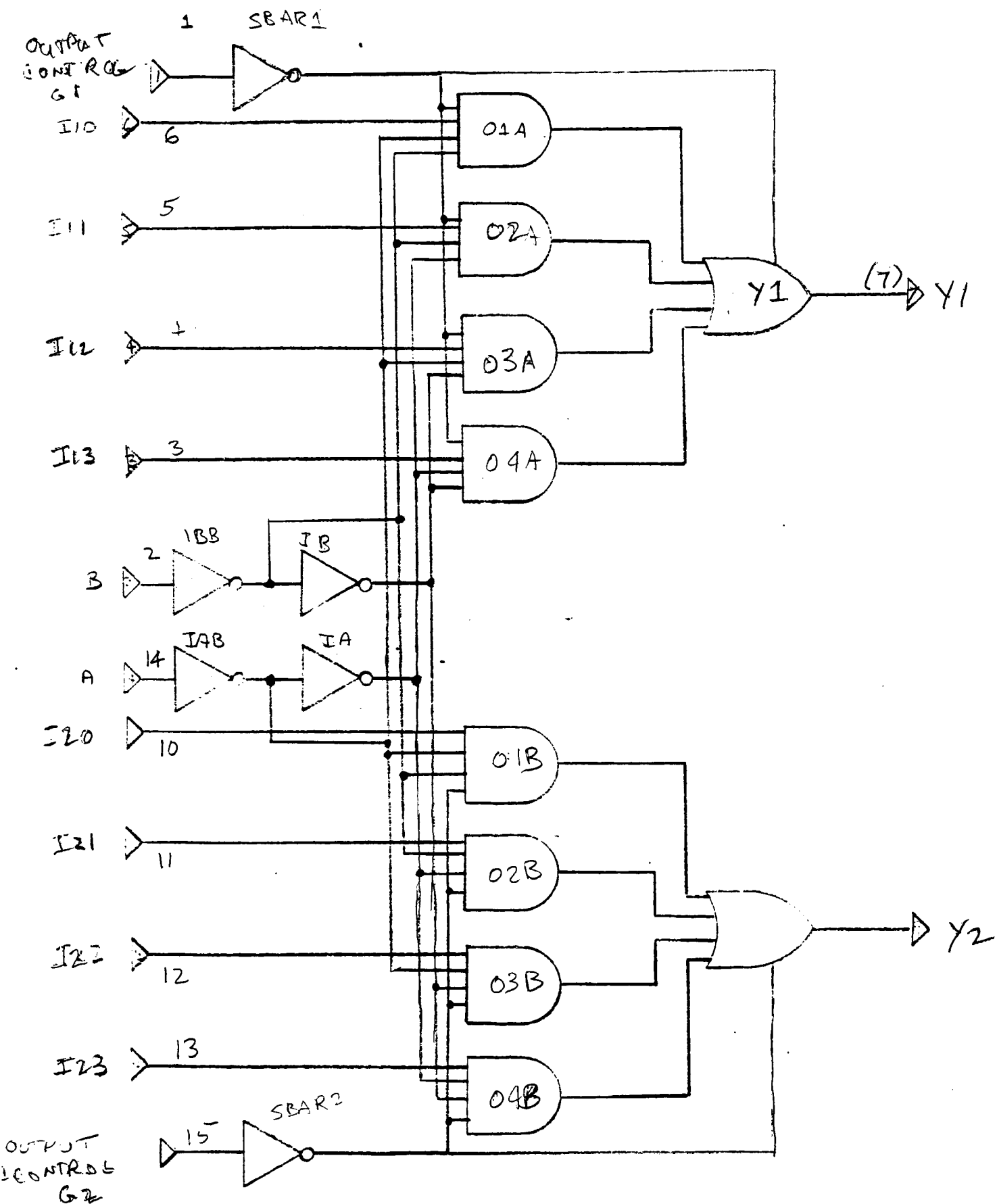
54LS158
FIGURE 11



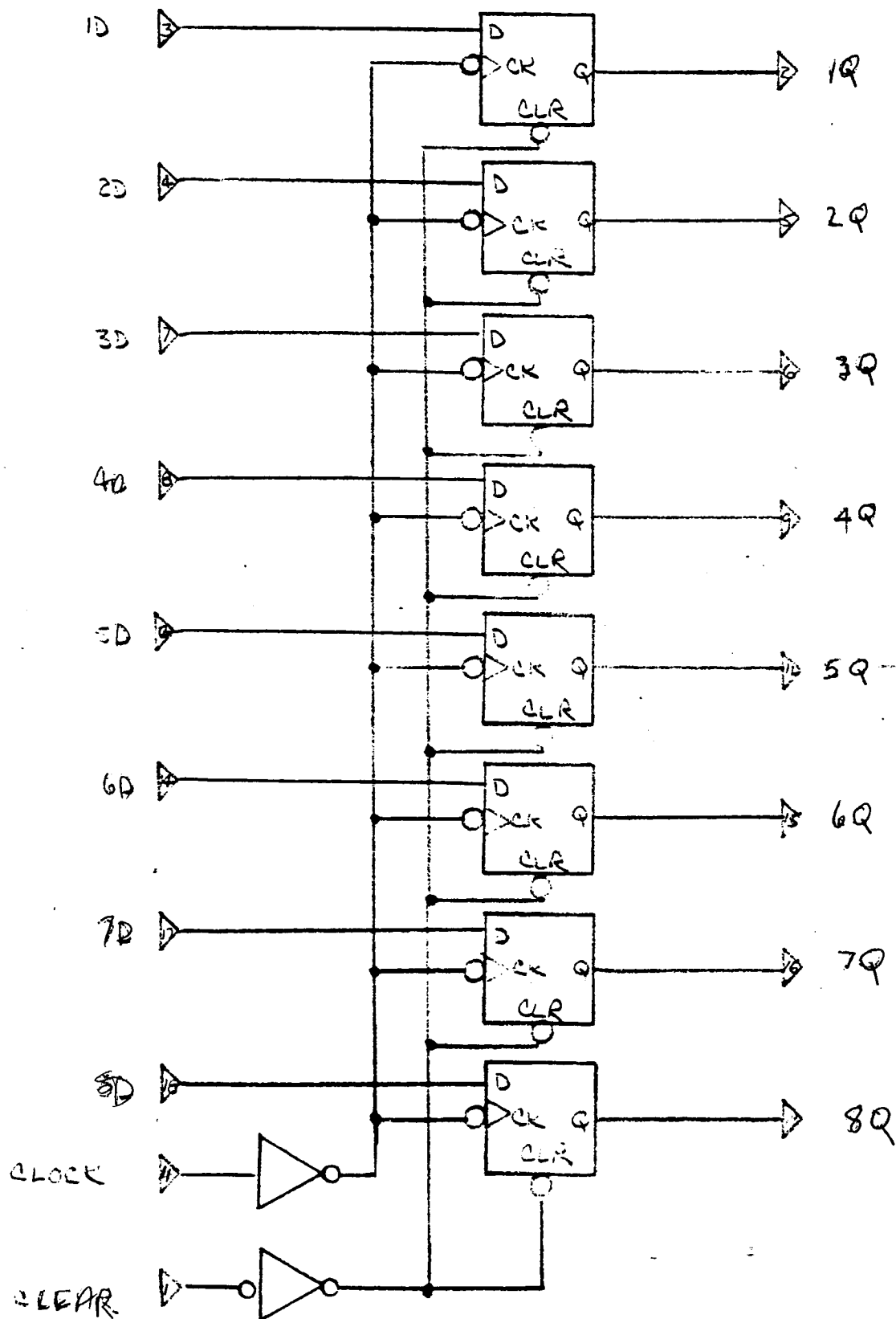
54LS169
FIGURE 12



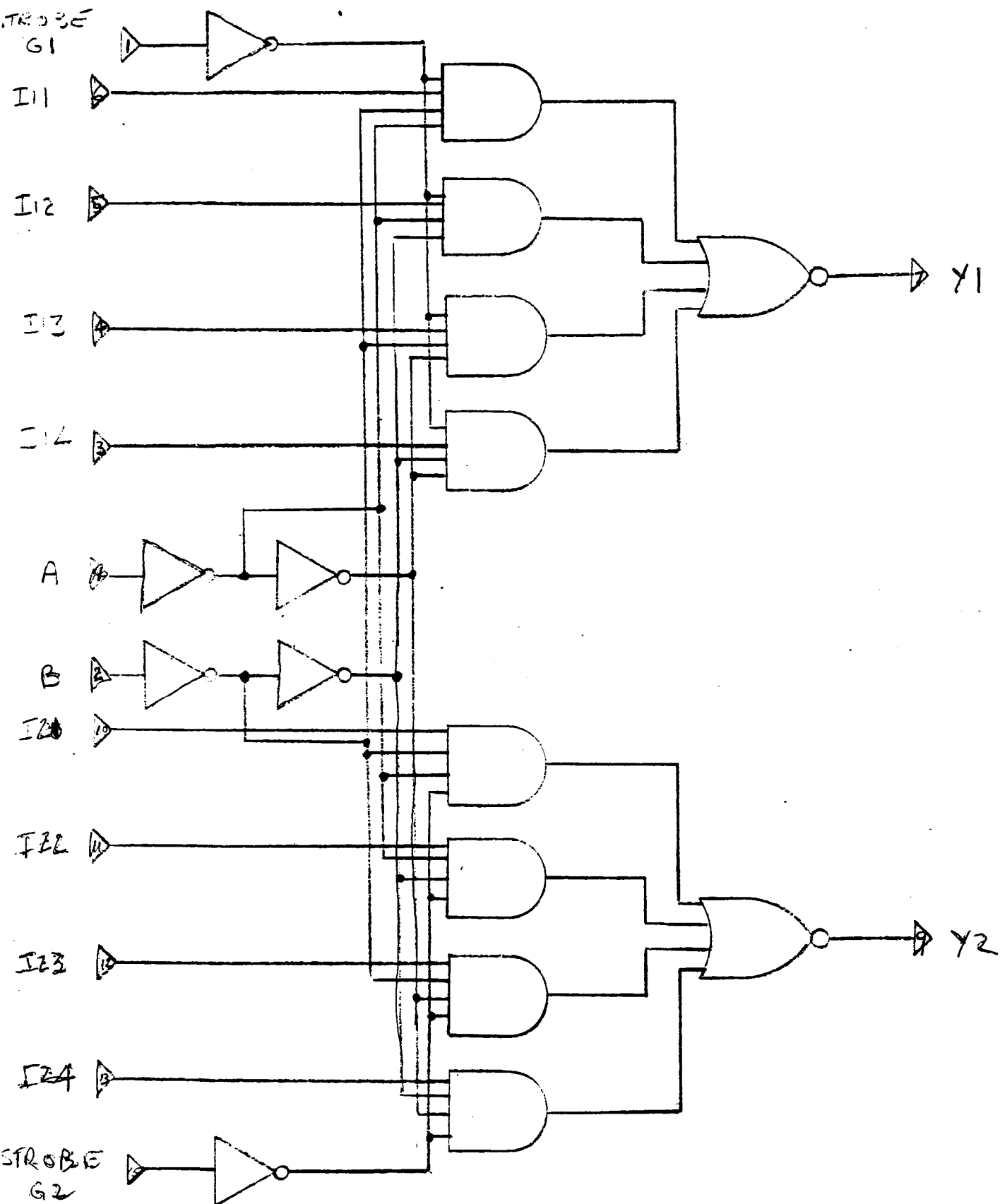
54LS175
FIGURE 13



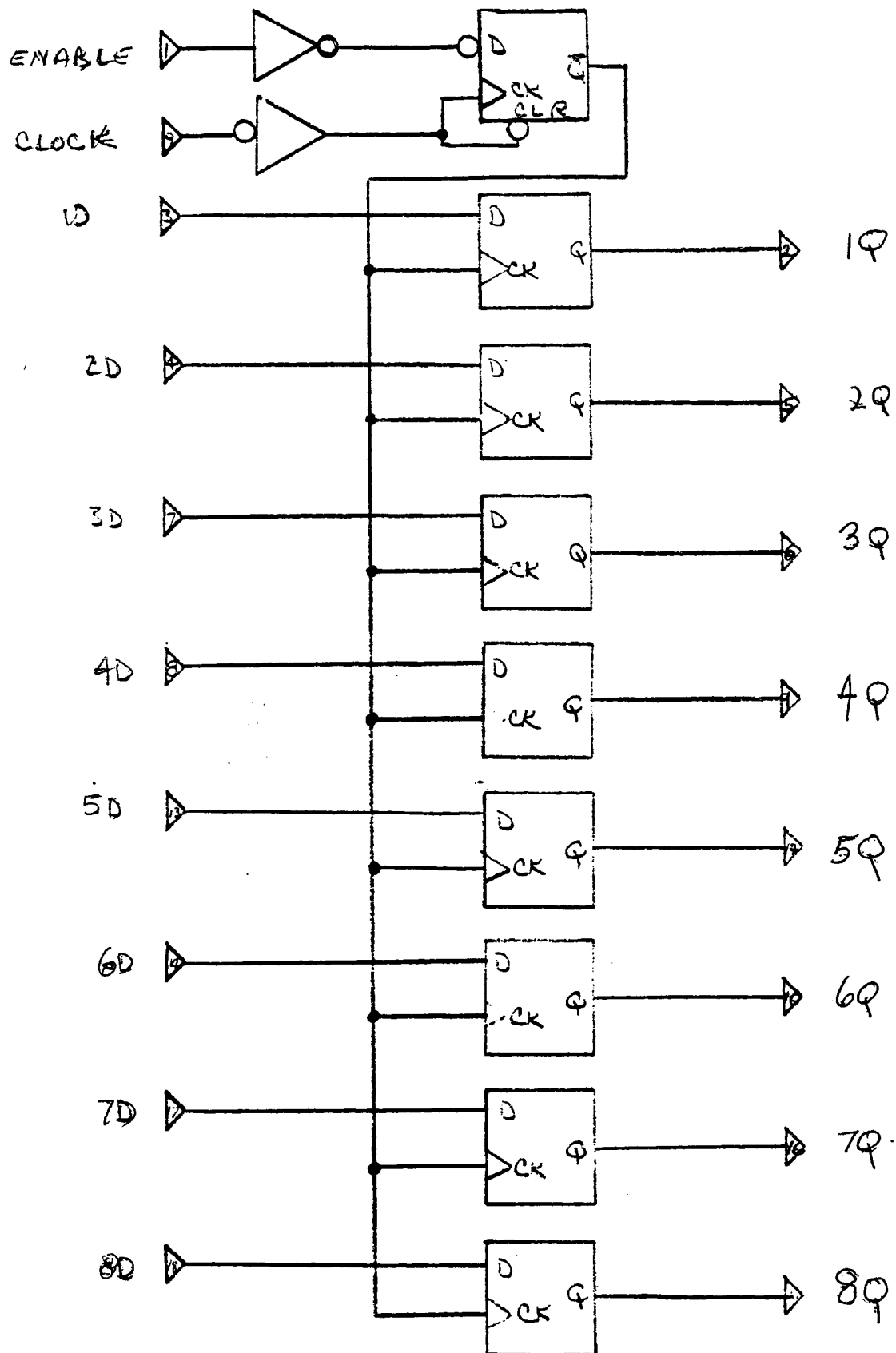
54LS253
FIGURE 14



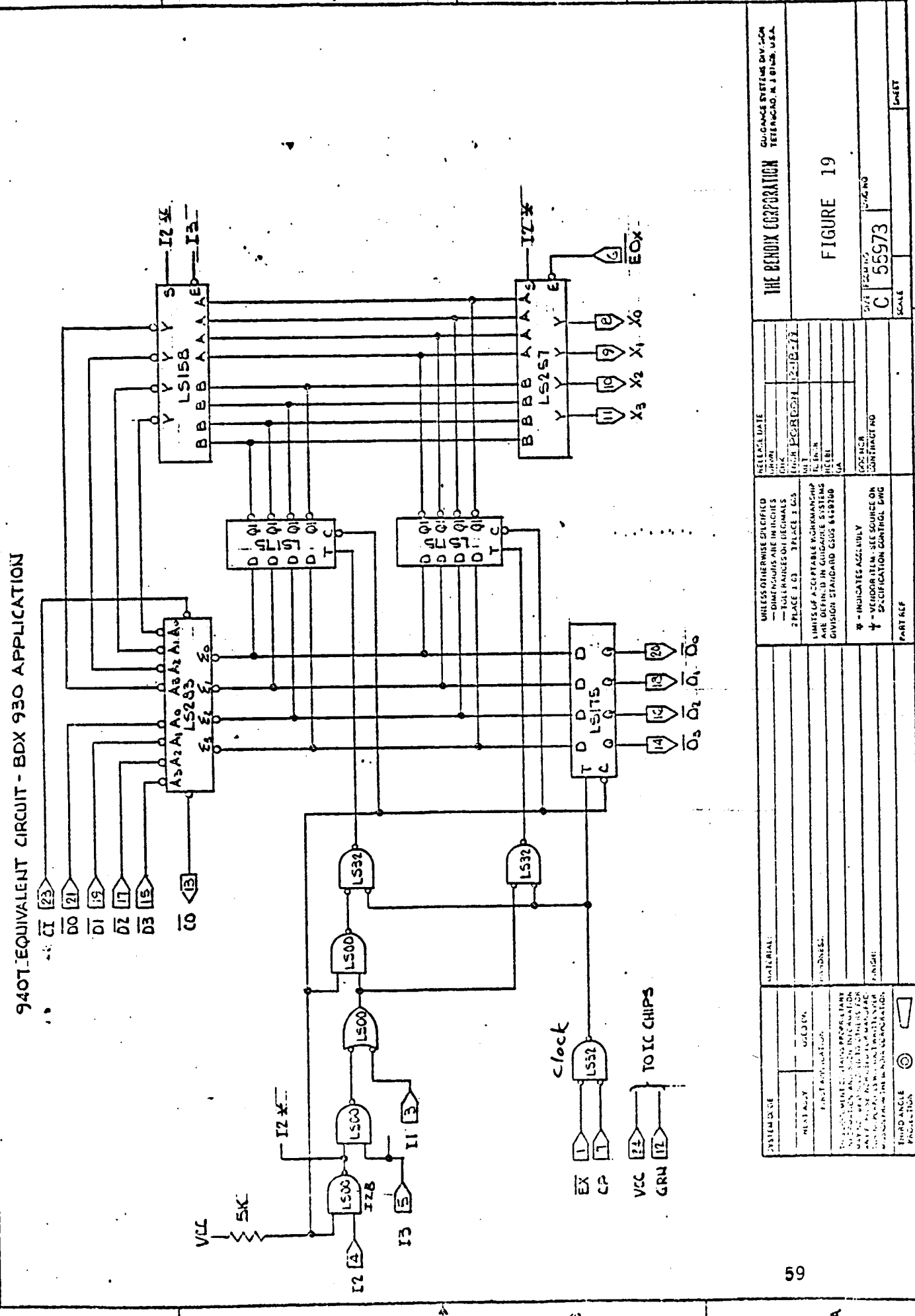
54LS273
FIGURE 15



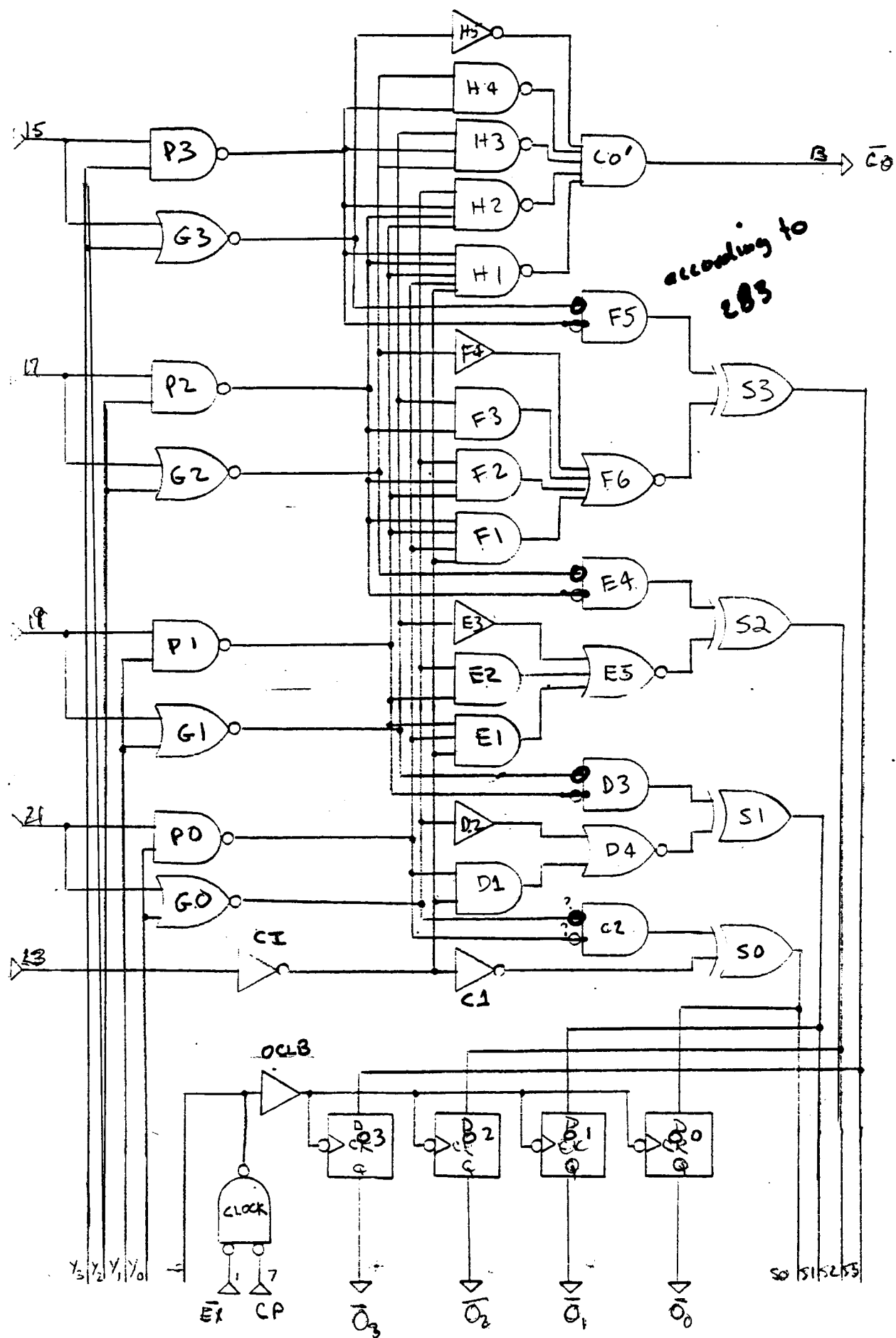
54LS352
FIGURE 16



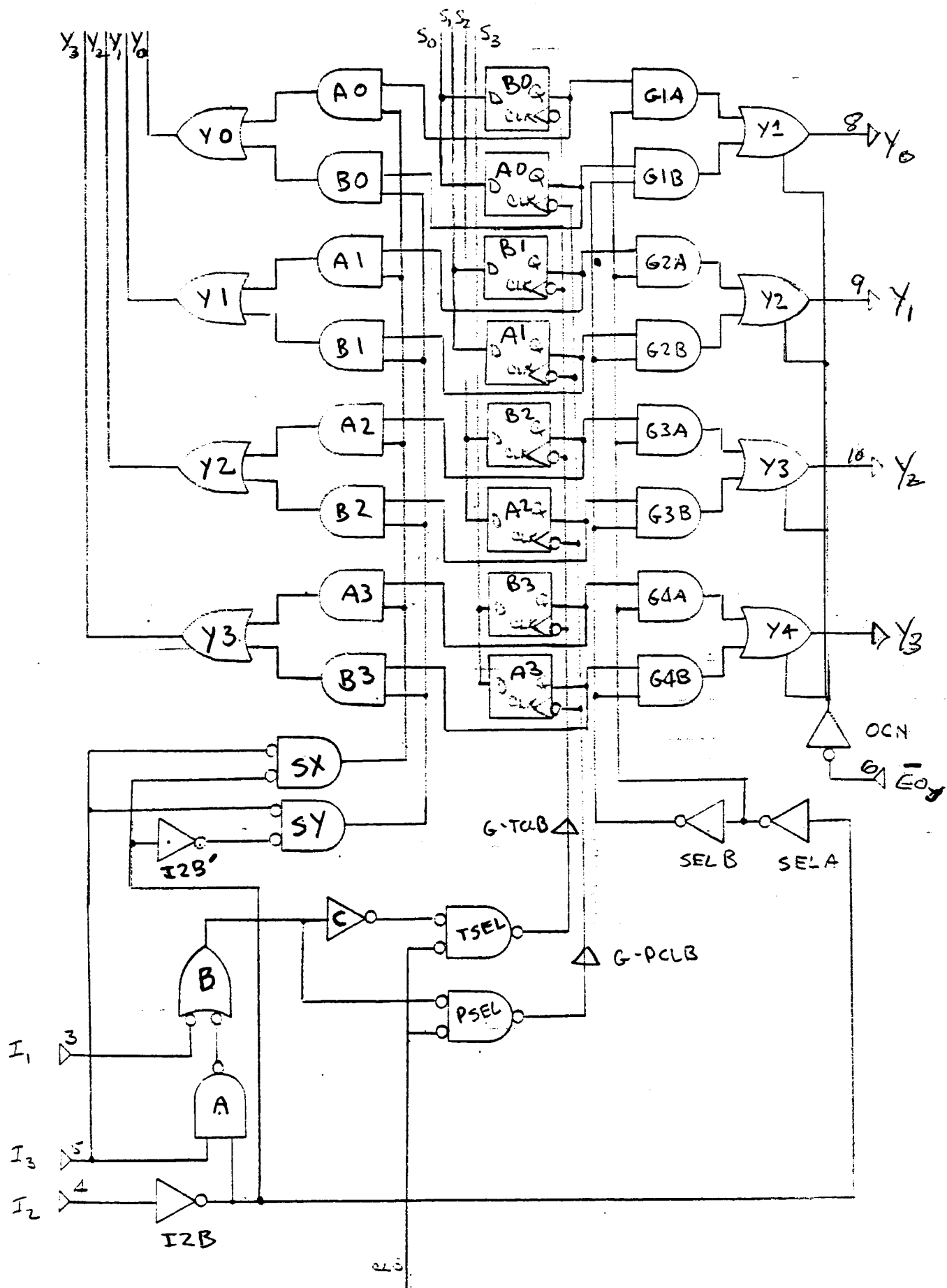
54LS377
FIGURE 18

[illegible]

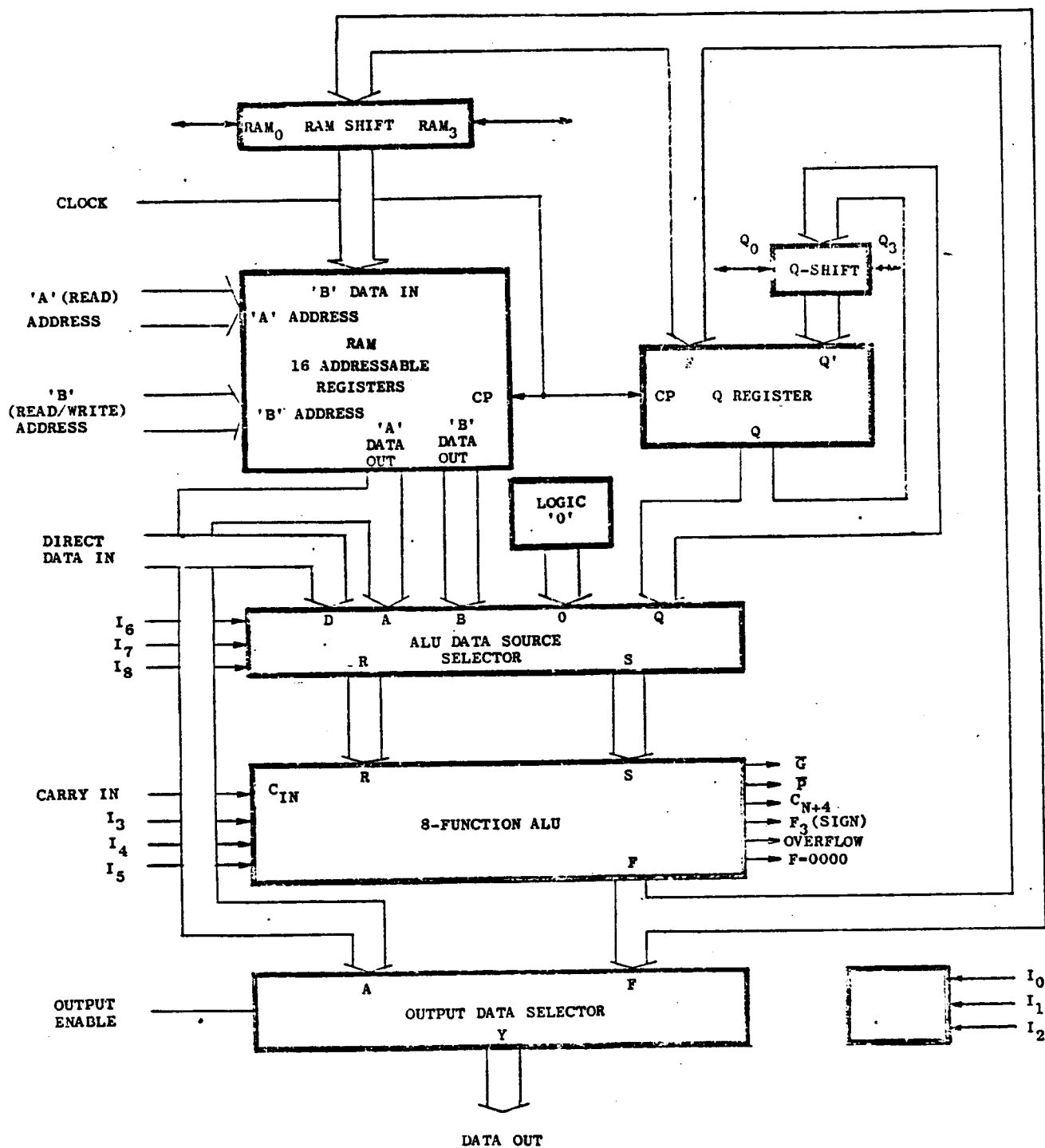
SYSTEM USE		MATERIAL		UNLESS OTHERWISE SPECIFIED — DIMENSIONS ARE IN INCHES — TOLERANCES ON DECIMALS		REPLACE DATE		THE BENDIX CORPORATION		GUIDANCE SYSTEMS DIV. 200 TETERBORO, N. J. 07812, U.S.A.	
NEXT ASY		SPEC. NO.		PLACE 1 03 PLACE 2 03		DATE FORWARDED		C 55973		DRAWING NO.	
FIRST APPLICATION		PARTS LIST		LIMITS ACCEPTABLE FOR MACHINING AND FINISH IN GUIDANCE SYSTEMS DIVISION STANDARD GS05 6410700		DATE		SCALE		DATE	
FOR DESIGN, DEVELOPMENT, PRODUCTION CONSTRUCTION AND QUALITY INSPECTION MAY BE USED BY OTHERS FOR ANY PURPOSES EXCEPT FOR MANUFACTURE OF THIS PART AND FOR REPRODUCTION OF THIS DRAWING FOR INFORMATION		FINISH		★ - INDICATES ASSEMBLY † - VENDOR ITEM. SEE SOURCE ON SPECIFICATION CONTROL ENG		COC. NO.		C 55973		DRAWING NO.	
THIRD ANGLE PROJECTION		PART REF		★ - INDICATES ASSEMBLY † - VENDOR ITEM. SEE SOURCE ON SPECIFICATION CONTROL ENG		COC. NO.		C 55973		DRAWING NO.	



9407 PART I
FIGURE 20

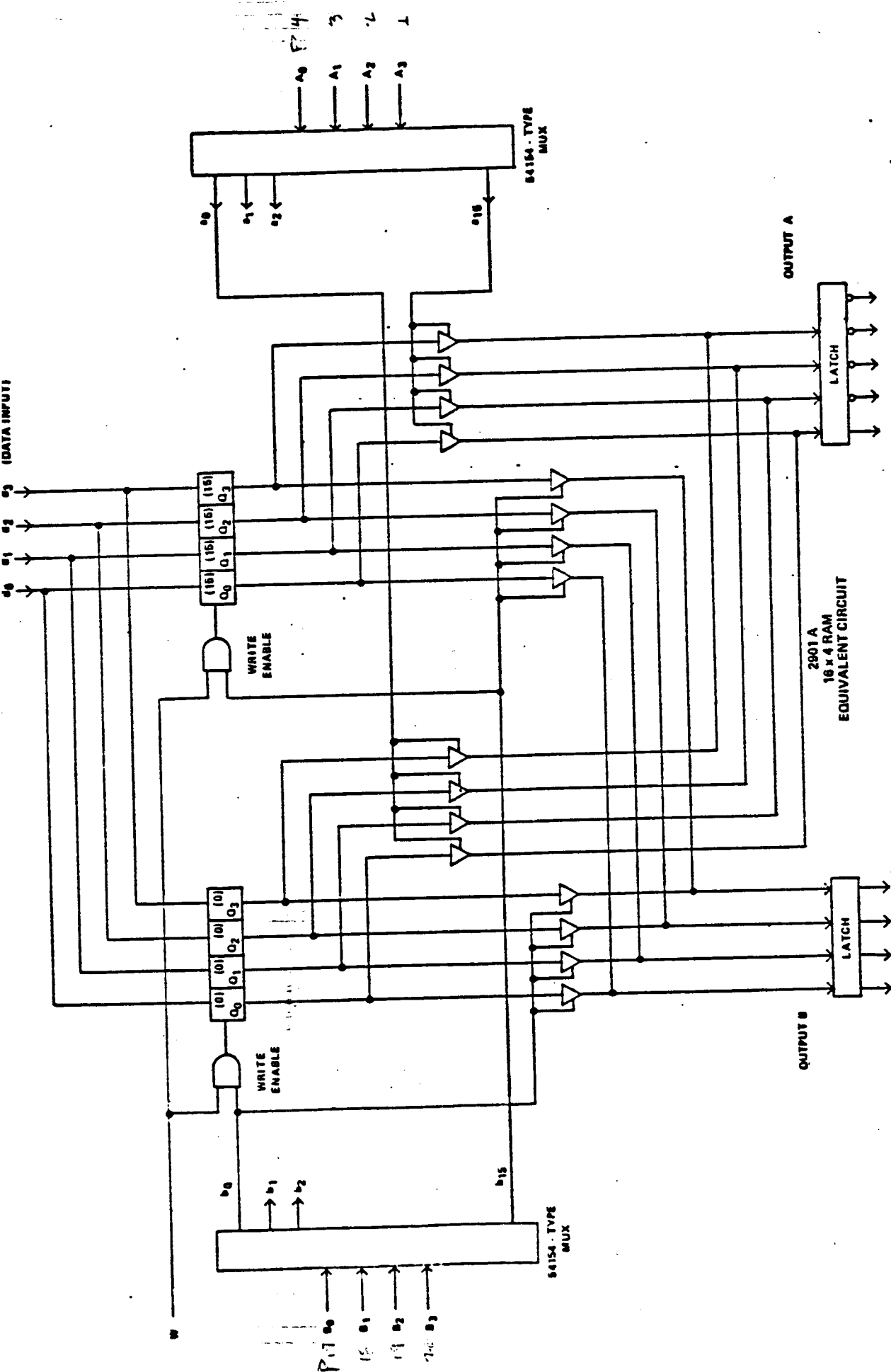


9407 PART II
FIGURE 20 (CONT'D)



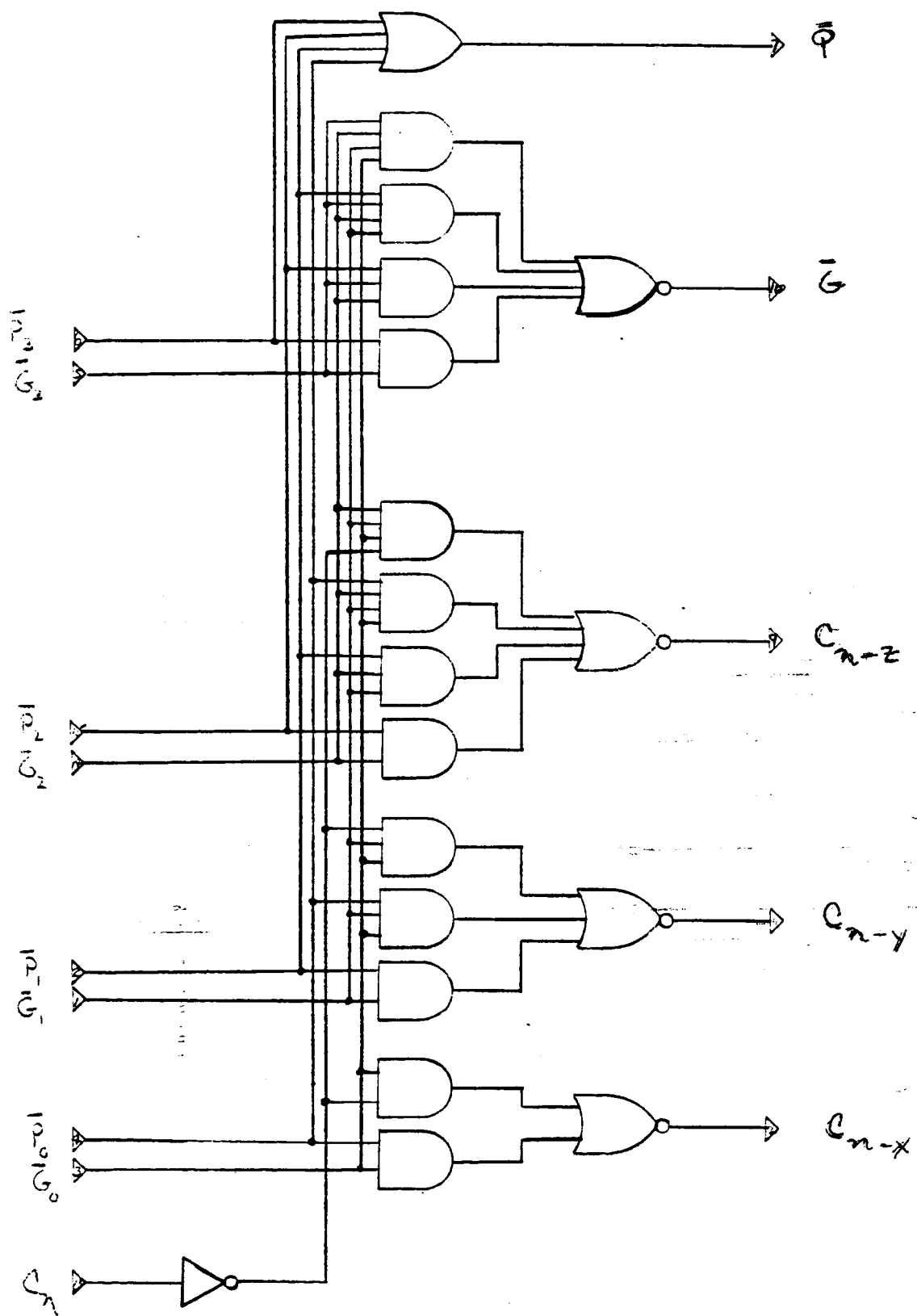
2901 4 - BIT SLICE
OVERVIEW
FIGURE 21





2901 RAM REPRESENTATION

FIGURE 23



AM2902A

FIGURE 24

8.0 INTERCONNECTION DESCRIPTION

The remainder of the BDX930 emulation description is contained in the following appendices.

- o Appendix D - BDX930 Processor Intercard
- o Appendix E - CPU Card
- o Appendix F - Timing and Control Card

9.0 EXTENSION TO A COMPLETE SIFT PROCESSOR

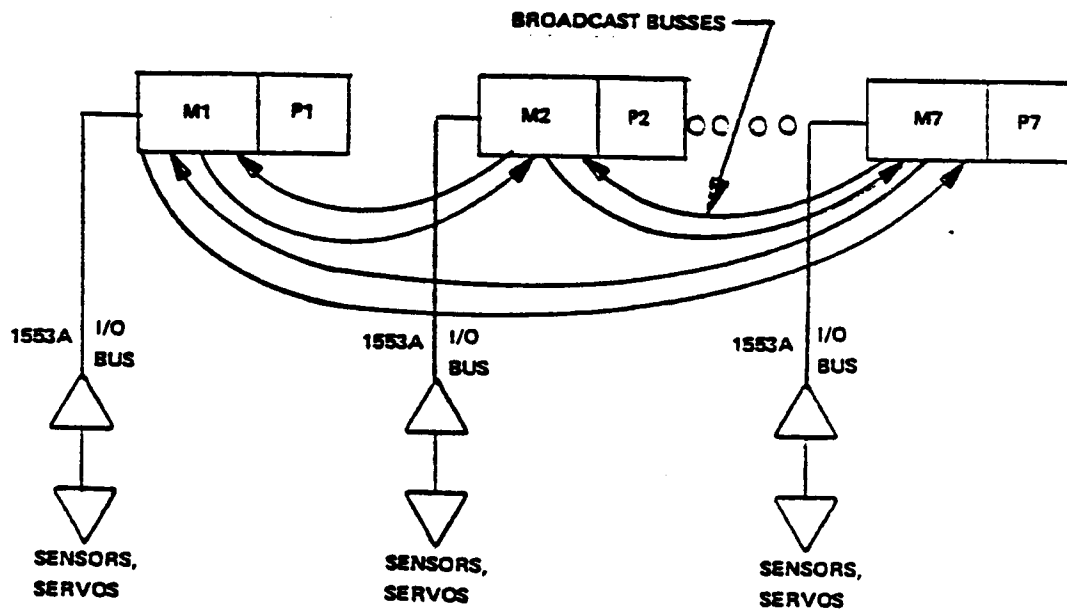
The objective of the present program is to eventually emulate the SIFT (Software Implemented Fault Tolerance) Computer System. While this report describes the BDX930 Processor, the next logical step is to describe a SIFT Processor with its associated I/O.

SIFT is an ultra-reliable computer system that is designed for flight critical aircraft control and avionics applications. It is based on a multiprocessor architecture that achieves fault tolerance by replicating computing tasks among processing units. Error detection and system reconfiguration are performed by software. The SIFT system is shown in Figure 23 in a 7-processor configuration. A single processor is shown in Figure 24.

In order to emulate a SIFT Processor, it is necessary to extend the present description to include the following cards.

- o J2 - Bus Processor Interface
- o J3 - Memory Interface and Control
- o J4 - Broadcast Receiver
- o J5 - 1553 Data Link
- o J7 & J8 - Main Memory, and
- o J9 - Remaining Timing and Control

This is approximately ten sheets of logic, compared to six sheets already simulated.



M1 M7 MEMORY
P1 P7 PROCESSOR

FIGURE 25 SIFT SYSTEM

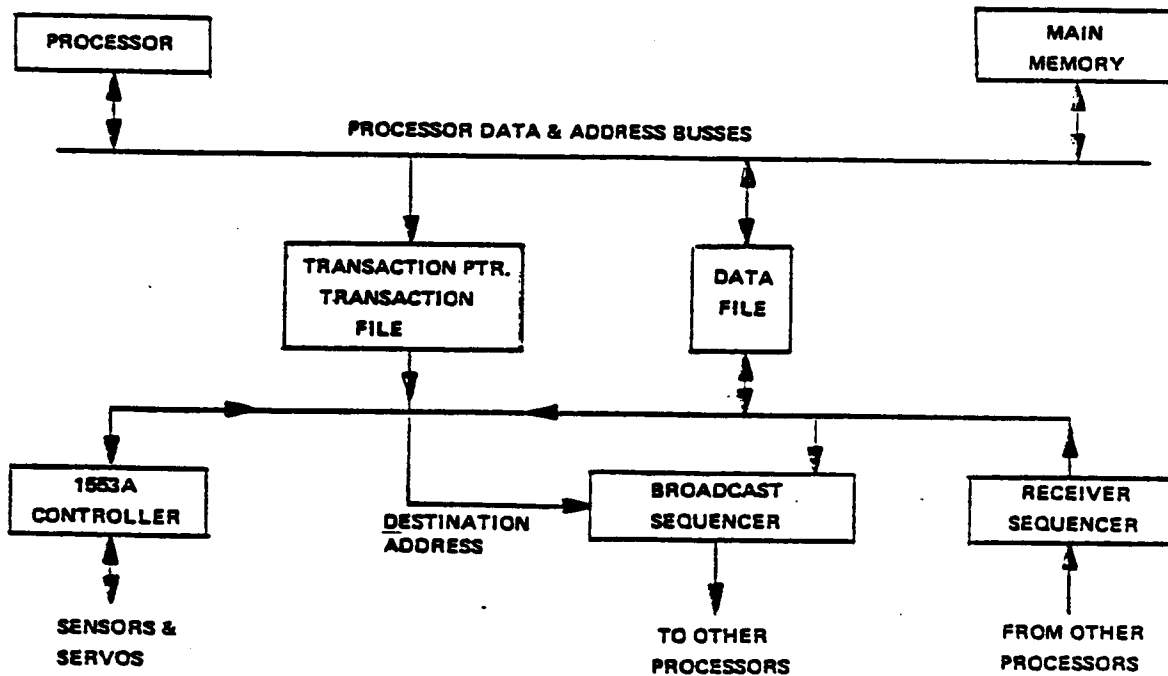


FIGURE 26 SIFT COMPUTER

10.0 REFERENCES

1. McGough, J., Swern, F., "Measurement of Fault Latency in a Digital Avionic Mini Processor", NASA CR-3462, NASA Langley Research Center, Hampton, VA, October 1981.

APPENDIX A

CPU CARD CHIP LIBRARY IN EMULATION

SYSTEM SYNTAX

\$ CHIP DEFINITION \$

TYPE: 54_S_00
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD 2-INPUT POSITIVE-NAND GATES.
UNUSED PINS: NONE
FUNCTIONS:

G-01(P-3) = NAND(P-1, P-2)
G-02(P-6) = NAND(P-4, P-5)
G-03(P-8) = NAND(P-9, P-10)
G-04(P-11) = NAND(P-12, P-13)

\$ END CHIP \$

TABLE 4

\$ CHIP DEFINITION \$

TYPE: 54-LS-00
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD 2-INPUT POSITIVE-NAND GATES.
UNUSED PINS: NONE
FUNCTIONS:

G-01(P-3) = NAND(P-1, P-2)
G-02(P-6) = NAND(P-4, P-5)
G-03(P-8) = NAND(P-9, P-10)
G-04(P-11) = NAND(P-12, P-13)

\$ END CHIP \$

TABLE 5

\$ CHIP DEFINITION \$

TYPE: 54-S-02
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD 2-INPUT POSITIVE-NOR GATES.
UNUSED PINS: NONE
FUNCTIONS:

G-01(P-1) = NOR(P-2, P-3)
G-02(P-4) = NOR(P-5, P-6)
G-03(P-10) = NOR(P-8, P-9)
G-04(P-13) = NOR(P-11, P-12)

\$ END CHIP \$

TABLE 6

S CHIP DEFINITION S

TYPE: 54_LS_02
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD 2-INPUT POSITIVE-NOR GATES.
UNUSED PINS: NONE
FUNCTIONS:

G-01(P-1) = NOR(P-2, P-3)
G-02(P-4) = NOR(P-5, P-6)
G-03(P-10) = NOR(P-8, P-9)
G-04(P-13) = NOR(P-11, P-12)

S END CHIP S

TABLE 7

\$ CHIP DEFINITION \$

TYPE: 54_S_04
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: HEX INVERTERS,
UNUSED PINS: NONE

FUNCTIONS:

G-01(P-2) = INV(P-1)
G-02(P-4) = INV(P-3)
G-03(P-6) = INV(P-5)
G-04(P-8) = INV(P-9)
G-05(P-10) = INV(P-11)
G-06(P-12) = INV(P-13)

\$ END CHIP \$

TABLE 8

S CHIP DEFINITION S

TYPE: 54_LS_08
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD 2-INPUT POSITIVE-AND GATES.
UNUSED PINS: NONE

FUNCTIONS:

G-01(P-3) = AND(P-1, P-2,)
G-02(P-6) = AND(P-4, P-5)
G-03(P-8) = AND(P-9, P-10,)
G-04(P-11) = AND(P-12, P-13)

S END CHIP S

TABLE 9

\$ CHIP DEFINITION \$

TYPE: 54_S_32
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD 2-INPUT POSITIVE-OR GATES.
UNUSED PINS: NONE

FUNCTIONS:

G-01(P-3) = OR(P-1, P-2,)
G-02(P-6) = OR(P-4, P-5,)
G-03(P-8) = OR(P-9, P-10)
G-04(P-11) = OR(P-12, P-13)

\$ END CHIP \$

TABLE 10

\$ CHIP DEFINITION \$

TYPE: 54_LS_86
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD 2-INPUT EXCLUSIVE-OR GATES.
UNUSED PINS: NONE

FUNCTIONS:

G-D1(P-3) = XOR(P-1, P-2)
G-D2(P-6) = XOR(P-4, P-5)
G-D3(P-8) = XOR(P-9, P-10)
G-D4(P-11) = XOR(P-12, P-13)

\$ END CHIP \$

TABLE 11

\$ CHIP DEFINITION \$

TYPE: 54_LS_113
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: DUAL GATED J - K FLIP-FLOP.
UNUSED PINS: NONE
FUNCTIONS:

FF-A(P-5, P-6) = J-K (PRESET=INV(P-4)
J=P-3
K=P-2
CLK_UP=P-1)

FF-B(P-9, P-8) = J-K (PRESET=INV(P-10)
J=P-11
K=P-12
CLK_UP=P-13

\$ END CHIP \$

TABLE 12

\$ CHIP DEFINITION \$

TYPE: 54_125
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD BUS BUFFER GATES WITH
THREE-STATE OUTPUTS.
UNUSED PINS: NONE

FUNCTIONS:

GTS-Y1(P-3)	= AND(P-2);	DIS_HIGH(P-1)
GTS-Y2(P-6)	= AND(P-5);	DIS_HIGH(P-4)
GTS-Y3(P-8)	= AND(P-9);	DIS_HIGH(P-10)
GTS-Y4(P-11)	= AND(P-12);	DIS_HIGH(P-13)

\$ END CHIP \$

TABLE 13

\$ CHIP DEFINITION \$

TYPE: 54-S-151

FAMILY: TTL

POWER: VCC = P-16, G-ND = P-8

DESCRIPTION: 1 - OF - 8 DATA SELECTORS/
MULTIPLEXERS.

UNUSED PINS: NONE

FUNCTIONS:

G-IAB = NOT(P-11)
G-IA = INV(G-IAB)

G-IBB = NOT(P-10)
G-IB = INV(G-IBB)

G-ICB = NOT(P-9)
G-IC = INV(G-ICB)

G-STB = INV(P-7)

G-00 = AND(G-STB, P-4, G-IAB, G-IBB, G-ICB)
G-01 = AND(G-STB, P-3, G-IA, G-IBB, G-ICB)
G-02 = AND(G-STB, P-2, G-IAB, G-IB, G-ICB)
G-03 = AND(G-STB, P-1, G-IA, G-IB, G-ICB)
G-04 = AND(G-STB, P-15, G-IAB, G-IBB, G-IC)
G-05 = AND(G-STB, P-14, G-IA, G-IBB, G-IC)
G-06 = AND(G-STB, P-13, G-IAB, G-IB, G-IC)
G-07 = AND(G-STB, P-12, G-IA, G-IB, G-IC)

G-W(P-6) = NOR(G-00, G-01, G-02, G-03, G-04,
G-05, G-06, G-07)

G-Y(P-5) = INV(G-W)

\$ END CHIP \$

TABLE 14

\$ -CHIP DEFINITION \$

TYPE: 54_LS_151

FAMILY: TTL

POWER: VCC = P-16, G-ND = P-8

DESCRIPTION: 1 - OF - 8 DATA SELECTORS/
MULTIPLEXERS.

UNUSED PINS: NONE

FUNCTIONS:

G-IAB = NOT(P-11)
G-IA = INV(G-IAB)

G-IBB = NOT(P-10)
G-IB = INV(G-IBB)

G-ICB = NOT(P-9)
G-IC = INV(G-ICB)

G-STB = INV(P-7)

G-00 = AND(G-STB, P-4, G-IAB, G-IBB, G-ICB)
G-01 = AND(G-STB, P-3, G-IA, G-IBB, G-ICB)
G-02 = AND(G-STB, P-2, G-IAB, G-IB, G-ICB)
G-03 = AND(G-STB, P-1, G-IA, G-IB, G-ICB)
G-04 = AND(G-STB, P-15, G-IAB, G-IBB, G-IC)
G-05 = AND(G-STB, P-14, G-IA, G-IBB, G-IC)
G-06 = AND(G-STB, P-13, G-IAB, G-IB, G-IC)
G-07 = AND(G-STB, P-12, G-IA, G-IB, G-IC)

G-W(P-6) = NOR(G-00, G-01, G-02, G-03, G-04,
G-05, G-06, G-07)

G-Y(P-5) = INV(G-W)

\$ END CHIP \$

TABLE 15

\$ CHIP DEFINITION \$

TYPE: 54_LS_153
 FAMILY: TTL
 POWER: VCC = P-16, GND = P-8
 DESCRIPTION: DUAL 1 OF 4 DATA SELECTORS/MULTIPLEXERS.
 UNUSED PINS: NONE

FUNCTIONS:

G-SBARA = INV(P-1)
 G-IAB = NOT(P-14)
 G-IA = INV(G-IAB)
 G-IBB = NOT(P-2)
 G-IB = INV(G-IBB)

 G-O1A = AND(G-SBARA, P-6, G-IAB, G-IBB)
 G-O2A = AND(G-SBARA, P-5, G-IA, G-IBB)
 G-O3A = AND(G-SBARA, P-4, G-IAB, G-IB)
 G-O4A = AND(G-SBARA, P-3, G-IA, G-IB)
 G-R1(P-7) = OR(G-O1A, G-O2A, G-O3A, G-O4A)

 G-SBARB = INV(P-15)
 G-O1B = AND(G-SBARB, P-10, G-IAB, G-IBB)
 G-O2B = AND(G-SBARB, P-11, G-IA, G-IBB)
 G-O3B = AND(G-SBARB, P-12, G-IAB, G-IB)
 G-O4B = AND(G-SBARB, P-13, G-IA, G-IB)
 G-R2(P-9) = OR(G-O1B, G-O2B, G-O3B, G-O4B)

 \$ END CHIP \$

TABLE 16

\$ CHIP DEFINITION \$

TYPE: 54-LS-158
 FAMILY: TTL
 POWER: VCC = P-16, GND = P-8
 DESCRIPTION: QUAD 2-LINE TO 1-LINE DATA
 SELECTORS/MULTIPLEXERS.
 UNUSED PINS: NONE

FUNCTIONS:

G-SB = NOT(P-1)
 G-SY = AND(INV(G-SB), INV(P-15))
 G-SX = AND(INV(P-1), INV(P-15))

G-A1 = AND(P-2, G-SX)
 G-B1 = AND(P-3, G-SY)
 G-O1(P-4) = NOR(G-A1, G-B1)

G-A2 = AND(P-5, G-SX)
 G-B2 = AND(P-6, G-SY)
 G-O2(P-7) = NOR(G-A2, G-B2)

G-A3 = AND(P-11, G-SX)
 G-B3 = AND(P-10, G-SY)
 G-O3(P-9) = NOR(G-A3, G-B3)

G-A4 = AND(P-14, G-SX)
 G-B4 = AND(P-13, G-SY)
 G-O4(P-12) = NOR(G-A4, G-B4)

\$ END CHIP \$

TABLE 17

S CHIP DEFINITION S
TYPE: 54_LS_169
FAMILY: TTL
POWER: VCC = P-16, GND = P-8
DESCRIPTION: SYNCHRONOUS 4-BIT UP/DOWN COUNTER.
UNUSED PINS: NONE
FUNCTIONS:

G-CK = NOT(P-2)
 G-UDI = NOT(P-1)
 G-LB = INV(P-9)
 G-L = NOT(G-LB)
 G-EP = INV(P-7)
 G-ET = INV(P-10)
 G-COUNT = AND(P-9, G-EP, G-ET)

G-QA1 = AND(G-UDI, FF-A)
 G-QA2 = AND(INV(G-UDI), FF-A')
 G-C1 = NOR(G-QA1, G-QA2)

G-QB1 = AND(G-UDI, FF-B)
 G-QB2 = AND(INV(G-UDI), FF-B')
 G-C2 = NOR(G-QB1, G-QB2)

G-QC1 = AND(G-UDI, FF-C)
 G-QC2 = AND(INV(G-UDI), FF-C')
 G-C3 = NOR(G-QC1, G-QC2)

G-QD1 = AND(G-UDI, FF-DX)
 G-QD2 = AND(INV(G-UDI), FF-DX')
 G-C4 = NOR(G-QD1, G-QD2)

G-A = NOT(G-COUNT)
 G-AB = AND(G-COUNT, FF-A')
 G-AA = AND(FF-A, G-A, G-L)
 G-IA = AND(G-LB, P-3)
 G-D1 = OR(G-AA, G-AB, G-IA)

G-B = NAND(G-COUNT, G-C1)
 G-BB = AND(G-COUNT, FF-B', G-C1)
 G-BA = AND(FF-B, G-B, G-L)
 G-IB = AND(G-LB, P-4)
 G-D2 = OR(G-BA, G-BB, G-IB)

G-C = NAND(G-COUNT, G-C1, G-C2)
 G-CB = AND(G-COUNT, FF-C', G-C1, G-C2)
 G-CA = AND(FF-C, G-C, G-L)
 G-IC = AND(G-LB, P-5)
 G-D3 = OR(G-CA, G-CB, G-IC)

G-DX = NAND(G-COUNT, G-C1, G-C2, G-C3)
 G-DB = AND(G-COUNT, FF-DX', G-C1, G-C2, G-C3)
 G-DA = AND(FF-DX, G-DX, G-L)
 G-ID = AND(G-LB, P-6)

TABLE 18

G-D4 = OR(G-DA, G-DB, G-ID)

G-CO(P-15) = NAND(INV(P-10), G-C1, G-C2, G-C3, G-C4)

FF-A(P-14) = D(CLK_UP=G-CK,
DATA=G-D1)

FF-B(P-13) = D(CLK_UP=G-CK,
DATA=G-D2)

FF-C(P-12) = D(CLK_UP=G-CK,
DATA=G-D3)

FF-DX(P-11) = D(CLK_UP=G-CK,
DATA=G-D4)

\$ END CHIP \$

TABLE 18 (CONT'D)

S CHIP DEFINITION S

TYPE: 54_LS_175

FAMILY: TTL

POWER: VCC = P-16, GND = P-8

DESCRIPTION: QUAD D-TYPE FLIP-FLOPS

UNUSED PINS: NONE

FUNCTIONS:

G-CLK=NOT(P-9)
G-CLR=NOT(INV(P-1))

FF-Q1(P-2,P-3) = D(CLEAR=INV(G-CLR),
DATA=P-4,
CLK_UP=INV(G-CLK))

FF-Q2(P-7,P-6) = D(CLEAR=INV(G-CLR),
DATA=P-5,
CLK_UP=INV(G-CLK))

FF-Q3(P-10,P-11) = D(CLEAR=INV(G-CLR),
DATA=P-12,
CLK_UP=INV(G-CLK))

FF-Q4(P-15,P-14) = DX(CLEAR=INV(G-CLR),
DATA=P-13,
CLK_UP=INV(G-CLK))

S END CHIP S

TABLE 19

\$ CHIP DEFINITION \$

TYPE: 54_LS_245
 FAMILY: TTL
 POWER: VCC = P-20, GND = P-10
 DESCRIPTION: OCTAL BUS TRANSCEIVERS
 WITH 3-STATE OUTPUTS.
 UNUSED PINS: NONE

FUNCTIONS:

G-B = AND(INV(P-1), INV(P-19))

G-OUTA1(P-2) = AND(G-OUTB1);	DIS_LOW(G-B)
G-OUTA2(P-3) = AND(G-OUTB2);	DIS_LOW(G-B)
G-OUTA3(P-4) = AND(G-OUTB3);	DIS_LOW(G-B)
G-OUTA4(P-5) = AND(G-OUTB4);	DIS_LOW(G-B)
G-OUTA5(P-6) = AND(G-OUTB5);	DIS_LOW(G-B)
G-OUTA6(P-7) = AND(G-OUTB6);	DIS_LOW(G-B)
G-OUTA7(P-8) = AND(G-OUTB7);	DIS_LOW(G-B)
G-OUTA8(P-9) = AND(G-OUTB8);	DIS_LOW(G-B)

G-A = AND(P-1, INV(-19))

G-OUTB1(P-18) = AND(G-OUTA1);	DIS_LOW(G-A)
G-OUTB2(P-17) = AND(G-OUTA2);	DIS_LOW(G-A)
G-OUTB3(P-16) = AND(G-OUTA3);	DIS_LOW(G-A)
G-OUTB4(P-15) = AND(G-OUTA4);	DIS_LOW(G-A)
G-OUTB5(P-14) = AND(G-OUTA5);	DIS_LOW(G-A)
G-OUTB6(P-13) = AND(G-OUTA6);	DIS_LOW(G-A)
G-OUTB7(P-12) = AND(G-OUTA7);	DIS_LOW(G-A)
G-OUTB8(P-11) = AND(G-OUTA8);	DIS_LOW(G-A)

\$ END CHIP \$

S CHIP DEFINITION S

TYPE: 54_LS_253
 FAMILY: TTL
 POWER: VCC = P-16, GND = P-8
 DESCRIPTION: DUAL 4-LINE-TO-1-LINE DATA
 SELECTORS/MULTIPLEXERS
 WITH 3-STATE OUTPUTS.
 UNUSED PINS: NONE

FUNCTIONS:

G-SBAR1 = INV(P-1)
 G-IAB = NOT(P-14)
 G-IA = INV(G-IAB)
 G-IBB = NOT(P-2)
 G-IB = INV(G-IBB)

G-O1A = AND(G-SBAR1, P-6, G-IAH, G-IBB)
 G-O2A = AND(G-SBAR1, P-5, G-IA, G-IBB)
 G-O3A = AND(G-SBAR1, P-4, G-IAB, G-IB)
 G-O4A = AND(G-SBAR1, P-3, G-IA, G-IB)
 GTS-1Y(P-7) = OR(G-O1A, G-O2A, G-O3A, G-O4A); DIS_LOW(G-SBAR1)

G-SBAR2 = INV(P-15)

G-O1B = AND(G-SBAR2, P-10, G-IAB, G-IBB)
 G-O2B = AND(G-SBAR2, P-11, G-IA, G-IBB)
 G-O3B = AND(G-SBAR2, P-12, G-IAB, G-IB)
 G-O4B = AND(G-SBAR2, P-14, G-IA, G-IB)
 GTS-2Y(P-9) = OR(G-O1B, G-O2B, G-O3B, G-O4B); DIS_LOW(G-SBAR2)

S END CHIP S

\$ CHIP DEFINITION \$

```

TYPE:                54_LS_273

FAMILY:              TTL

POWER:               VCC = P-20, G-ND = P-10

DESCRIPTION:         OCTAL D-TYPE FLIP-FLOPS,

UNUSED PINS:         NONE

FUNCTIONS:

G-CLB = NOT(P-11)
G-CLR = NOT(INV(P-1))

FF-D1(P-2) = D(CLEAR=INV(G-CLR),
               DATA=P-3,
               CLK_UP=INV(G-CLB))

FF-D2(P-5) = D(CLEAR=INV(G-CLR),
               DATA=P-4,
               CLK_UP=INV(G-CLB))

FF-D3(P-6) = D(CLEAR=INV(G-CLR),
               DATA=P-7,
               CLK_UP=INV(G-CLB))

FF-D4(P-9) = D(CLEAR=INV(G-CLR),
               DATA=P-8,
               CLK_UP=INV(G-CLB))

FF-D5(P-12) = D(CLEAR=INV(G-CLR),
                DATA=P-13,
                CLK_UP=INV(G-CLB))

FF-D6(P-15) = D(CLEAR=INV(G-CLR),
                DATA=P-14,
                CLK_UP=INV(G-CLB))

FF-D7(P-16) = D(CLEAR=INV(G-CLR),
                DATA=P-17,
                CLK_UP=INV(G-CLB))

FF-D8(P-19) = D(CLEAR=INV(G-CLR),
                DATA=P-18,
                CLK_UP=INV(G-CLB))

$ END CHIP $

```

\$ CHIP DEFINITION \$

TYPE: 54_S_288
 FAMILY TTL
 POWER: VCC = P-16, GND = P-8
 DESCRIPTION: 32 WORDS BY 8 BIT PROM
 WITH 3-STATE OUTPUTS.
 UNUSED PINS: NONE

FUNCTIONS:

ROMTS-D1(P-1) = ADDRESS(P-10, P-11, P-12, P-13, P-14);
 DIS_HIGH(P-15)
 ROMTS-D2(P-2) = ADDRESS(P-10, P-11, P-12, P-13, P-14);
 DIS_HIGH(P-15)
 ROMTS-D3(P-3) = ADDRESS(P-10, P-11, P-12, P-13, P-14);
 DIS_HIGH(P-15)
 ROMTS-D4(P-4) = ADDRESS(P-10, P-11, P-12, P-13, P-14);
 DIS_HIGH(P-15)
 ROMTS-D5(P-5) = ADDRESS(P-10, P-11, P-12, P-13, P-14);
 DIS_HIGH(P-15)
 ROMTS-D6(P-6) = ADDRESS(P-10, P-11, P-12, P-13, P-14);
 DIS_HIGH(P-15)
 ROMTS-D7(P-7) = ADDRESS(P-10, P-11, P-12, P-13, P-14);
 DIS_HIGH(P-15)
 ROMTS-D8(P-9) = ADDRESS(P-10, P-11, P-12, P-13, P-14);
 DIS_HIGH(P-15)

\$ END CHIP \$

\$ CHIP DEFINITION \$

TYPE: 54-LS-352

FAMILY: TTL

POWER: VCC = P-16, GND = P-8

DESCRIPTION: DUAL 4-LINE-TO-1-LINE DATA
SELECTORS/MULTIPLEXERS.

UNUSED PINS: NONE

FUNCTIONS:

G-SBAR1 = INV(P-1)
G-IAB = NOT(P-14)
G-IA = INV(G-IAB)
G-IBB = NOT(P-2)
G-IB = INV(G-IBB)

G-O1A = AND(G-SBAR1, P-6, G-IAB, G-IBB)
G-O2A = AND(G-SBAR1, P-5, G-IA, G-IBB)
G-O3A = AND(G-SBAR1, P-4, G-IAB, G-IB)
G-O4A = AND(G-SBAR1, P-3, G-IA, G-IB)
G-OUT1(P-7) = NOR(G-O1A, G-O2A, G-O3A, G-O4A)

G-SBAR2 = INV(P-15)

G-O1B = AND(G-SBAR2, P-10, G-IAB, G-IBB)
G-O2B = AND(G-SBAR2, P-11, G-IA, G-IBB)
G-O3B = AND(G-SBAR2, P-12, G-IAB, G-IB)
G-O4B = AND(G-SBAR2, P-13, G-IA, G-IB)
G-OUT2(P-9) = NOR(G-O1B, G-O2B, G-O3B, G-O4B)

\$ END CHIP \$

TABLE 24.

S CHIP DEFINITION S

TYPE: 54_LS-367
 FAMILY: TTL
 POWER: VCC = P-16, GND = P-8
 DESCRIPTION: HEX BUS DRIVERS WITH 3-STATE OUTPUTS.
 UNUSED PINS: NONE

FUNCTIONS:

G-A1 = AND(INV(P-15))
 G-A2 = AND(INV(P-1))
 GTS-01(P-3) = AND(P-2); DIS_HIGH(G-A2)
 GTS-02(P-5) = AND(P-4); DIS_HIGH(G-A2)
 GTS-03(P-7) = AND(P-6); DIS_HIGH(G-A2)
 GTS-04(P-9) = AND(P-10); DIS_HIGH(G-A2)
 GTS-05(P-11) = AND(P-12); DIS_HIGH(G-A1)
 GTS-06(P-13) = AND(P-14); DIS_HIGH(G-A1)

S END CHIP S

\$ CHIP DEFINITION \$

```

TYPE:          54_LS_374

FAMILY:        TTL

POWER:         VCC = P-20, GND = P-10

DESCRIPTION:    OCTAL D-TYPE TRANSPARENT,
                LATCHES, AND EDGE-TRIGGERED
                FLIP-FLOPS WITH 3-STATE OUTPUTS.

UNUSED PINS:    NONE

FUNCTIONS:

G-OCB = INV(P-1
G-CLB = NOT(P-11)

FF-Q1 = D(DATA=P-3,
           CLK_UP=G-CLB)
G-Q1(P-2) = AND(INV(FF-Q1')); DIS_LOW(G-OCB)

FF-Q2 = D(DATA=P-4,
           CLK_UP=G-CLB)
G-Q2(P-5) = AND(INV(FF-Q2')); DIS_LOW(G-OCB)

FF-Q3 = D(DATA=P-7,
           CLK_UP=G-CLB)
G-Q3(P-6) = AND(INV(FF-Q3')); DIS_LOW(G-OCB)

FF-Q4 = D(DATA=P-8,
           CLK_UP=G-CLB)
G-Q4(P-9) = AND(INV(FF-Q4')); DIS_LOW(G-OCB)

FF-Q5 = D(DATA=P-13,
           CLK_UP=G-CLB)
G-Q5(P-12) = AND(INV(FF-Q5')); DIS_LOW(G-OCB)

FF-Q6 = D(DATA=P-14,
           CLK_UP=G-CLB)
G-Q6(P-15) = AND(INV(FF-Q6')); DIS_LOW(G-OCB)

FF-Q7 = D(DATA=P-17,
           CLK_UP=G-CLB)
G-Q7(P-16) = AND(INV(FF-Q7')); DIS_LOW(G-OCB)

FF-Q8 = D(DATA=P-18,
           CLK_UP=G-CLB)
G-Q8(P-19) = AND(INV(FF-Q8')); DIS_LO(G-OCB)

$ END CHIP $

```

TABLE 26

```

$ CHIP DEFINITION $

TYPE:          25_LS_377

FAMILY:        TTL

POWER:         VCC = P-20, GND = P-10

DESCRIPTION:    OCTAL D-TYPE FLIP-FLOPS.

UNUSED PINS:    NONE

FUNCTIONS:

G-CLB = NOT(P-11)
G-B = INV(P-1)

FF-ENBL = D(DATA=INV(G-B),
             CLK_UP=G-CLB,
             CLEAR=INV(G-CLB))

FF-Q1(P-2) = D(DATA=P-3,
               CLK_UP=FF-ENBL)

FF-Q2(P-5) = D(DATA=P-4,
               CLK_UP=FF-ENBL)

FF-Q3(P-6) = D(DATA=P-7,
               CLK_UP=FF-ENBL)

FF-Q4(P-9) = D(DATA=P-8,
               CLK_UP=FF-ENBL)

FF-Q5(P-12) = D(DATA=P-13,
                CLK_UP=FF-ENBL)

FF-Q6(P-15) = D(DATA=P-14,
                CLK_UP=FF-ENBL)

FF-Q7(P-16) = D(DATA=P-17,
                CLK_UP=FF-ENBL)

FF-Q8(P-19) = D(DATA=P-18,
                CLK_UP=FF-ENBL)

$ END CHIP $

```

TABLE 27

S CHIP DEFINITION S

TYPE: 54_S_472
 FAMILY: TTL
 POWER: VCC = P-20, GND = P-10
 DESCRIPTION: 512 WORDS BY 8 BIT PROM
 WITH 3-STATE OUTPUTS.
 UNUSED PINS: NONE

FUNCTIONS:

ROMTS-D1(P-6) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
 P-18, P-19); DIS_HIGH(P-15)
 ROMTS-D2(P-7) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
 P-18, P-19); DIS_HIGH(P-15)
 ROMTS-D3(P-8) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
 P-18, P-19); DIS_HIGH(P-15)
 ROMTS-D4(P-9) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
 P-18, P-19); DIS_HIGH(P-15)
 ROMTS-D5(P-11) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
 P-18, P-19); DIS_HIGH(P-15)
 ROMTS-D6(P-12) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
 P-18, P-19); DIS_HIGH(P-15)
 ROMTS-D7(P-13) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
 P-18, P-19); DIS_HIGH(P-15)
 ROMTS-D8(P-14) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
 P-18, P-19); DIS_HIGH(P-15)

S END CHIP S

```

$ CHIP DEFINITION $

TYPE:          54_LS_472

FAMILY:        TTL

POWER:         VCC = P-20, GND = P-10

DESCRIPTION:    512 WORDS BY 8 BIT PROM
                WITH 3-STATE OUTPUTS.

UNUSED PINS:    NONE

FUNCTIONS:

ROMTS-D1(P-6)  = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
                    P-18, P-19); DIS_HIGH(P-15)

ROMTS-D2(P-7)  = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
                    P-18, P-19); DIS_HIGH(P-15)

ROMTS-D3(P-8)  = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
                    P-18, P-19); DIS_HIGH(P-15)

ROMTS-D4(P-9)  = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
                    P-18, P-19); DIS_HIGH(P-15)

ROMTS-D5(P-11) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
                    P-18, P-19); DIS_HIGH(P-15)

ROMTS-D6(P-12) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
                    P-18, P-19); DIS_HIGH(P-15)

ROMTS-D7(P-13) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
                    P-18, P-19); DIS_HIGH(P-15)

ROMTS-D8(P-14) = ADDRESS(P-1, P-2, P-3, P-4, P-5, P-16, P-17,
                    P-18, P-19); DIS_HIGH(P-15)

$ END CHIP $

```

TABLE 29

S CHIP DEFINITION S

TYPE: 9407

FAMILY: TTL

POWER: VCC = P-24, GND = P-12

DESCRIPTION: MEMORY ADDRESS PROCESSOR,

UNUSED PINS: NONE

FUNCTIONS:

G-OCN = INV(P-6)

G-I2B = NOT(P-4)

G-SELA = NOT(G-I2B)

G-SELB = INV(G-SELA)

G-G1A = AND(FF-A0, G-SELA)

G-G1B = AND(FF-B0, G-SELB)

GTS-Y1(P-8) = OR(G-G1A, G-G1B);DIS_LOW(G-OCN)

G-G2A = AND(FF-A1, G-SELA)

G-G2B = AND(FF-B1, G-SELB)

GTS-Y2(P-9) = OR(G-G2A, G-G2B);DIS_LOW(G-OCN)

G-G3A = AND(FF-A2, G-SELA)

G-G3B = AND(FF-B2, G-SELB)

GTS-Y3(P-10) = OR(G-G3A, G-G3B);DIS_LOW(G-OCN)

G-G4A = AND(FF-A3, G-SELA)

G-G4B = AND(FF-B3, G-SELB)

GTS-Y4(P-11) = OR(G-G4A, G-G4B);DIS_LOW(G-OCN)

G-A = NAND(P-5, G-I2B)

G-B = NAND(G-A, P-3)

G-C = NOT(G-B)

G-CLOCK = OR(P-7, P-1)

G-TSEL = OR(G-CLOCK, G-C)

G-PSEL = OR(G-CLOCK, G-B)

G-I2B' = NOT(G-I2B)

G-SY = AND(INV(G-I2B'), INV(P-5))

G-SX = AND(INV(G-I2B), INV(P-5))

G-A0 = AND(FF-A0, G-SX)

G-B0 = AND(FF-B0, G-SY)

G-Y0 = NOR(G-A1, G-B1)

G-A1 = AND(FF-A1, G-SX)

G-B1 = AND(FF-B1, G-SY)

G-Y1 = NOR(G-A2, G-B2)

G-A2 = AND(FF-A2, G-SX)

G-B2 = AND(FF-B2, G-SY)

G-Y4 = NOR(G-A3, G-B3)

G-A3 = AND(FF-A3, G-SX)

G-B3 = AND(FF-B3, G-SY)

G-Y3 = NOR(G-A4, G-B4)

G-G0 = NOR(P-21, G-Y0)

G-P0 = NAND(P-21, G-Y0)

G-G1 = NOR(P-19, G-Y1)

TABLE 30

G-P1 = NAND(P-19, G-Y1)
 G-G2 = NOR(P-17, G-Y2)
 G-P2 = NAND(P-17, G-Y2)
 G-G3 = NOR(P-15, G-Y3)
 G-P3 = NAND(P-15, G-Y3)

G-C1 = NOT(P-23)
 G-C1 = NOT(G-C1)
 G-C2 = AND(INV(G-G0), G-P0)
 G-S0 = XOR(G-C1, G-C2)
 G-D2 = AND(G-G0)
 G-D1 = AND(G-P0, G-C1)
 G-D4 = NOR(G-D1, G-D2)
 G-D3 = AND(INV(G-G1), G-P1)
 G-S1 = XOR(G-D3, G-D4)
 G-E3 = AND(G-G1)
 G-E2 = AND(G-G0, G-P1)
 G-E1 = AND(G-P1, G-P0, G-C1)
 G-E5 = NOR(G-E1, G-E2, G-E3)
 G-E4 = AND(INV(G-G2), G-P2)
 G-S2 = XOR(G-E4, G-E5)
 G-F4 = AND(G-G2)
 G-F3 = AND(G-G1, G-P2)
 G-F2 = AND(G-P2, G-P1, G-G0)
 G-F1 = AND(P-P2, P-P1, P-P0, P-C1)
 G-F6 = NOR(G-F1, G-F2, G-F3, G-F4)
 G-F5 = AND(INV(G-G3), G-P3)
 G-S3 = XOR(G-F5, G-F6)
 G-H5 = INV(G-G3)
 G-H4 = NAND(G-G2, G-P3)
 G-H3 = NAND(G-P3, G-P2, G-G1)
 G-H2 = NAND(G-P3, G-P2, G-P1, G-G0)
 G-H1 = NAND(G-P3, G-P2, G-P1, G-P0, G-C1)
 G-CU'(P-13) = AND(G-H1, G-H2, G-H3, G-H4, G-H5)

G-CLB = AND(G-PSEL)
 FF-A0 = D(DATA = G-S0,
 CLK_DOWN = G-PCLB)
 FF-A1 = D(DATA = G-S1,
 CLK_DOWN = G-PCLB)
 FF-A2 = D(DATA = G-S2,
 CLK_DOWN = G-PCLB)
 FF-A3 = D(DATA = G-S3,
 CLK_DOWN = G-PCLB)

G-TCLB = AND(G-TSEL)
 FF-B0 = D(DATA = G-S0,
 CLK_DOWN = G-TCLB)
 FF-B1 = D(DATA = G-S1,
 CLK_DOWN = G-TCLB)
 FF-B2 = D(DATA = G-S2,
 CLK_DOWN = G-TCLB)
 FF-B3 = D(DATA = G-S3,
 CLK_DOWN = G-TCLB)

G-OCLB = AND(CLOCK)
 FF-O0(;P-20) = D(DATA = G-S0,
 CLK_DOWN = G-OCLB)
 FF-O1(;P-18) = D(DATA = G-S1,

TABLE 30 (CONT'D)

```
CLK_DOWN = G-OCLB)
FF-02(;P=16) = D(DATA = G-S2,
CLK_DOWN = G-OCLB)
FF-03(;P=14) = D(DATA = G-S3,
CLK_DOWN = G-OCLB)
```

```
$ END CHIP $
```

TABLE 30 (CONT'D)

S CHIP DEFINITION S

TYPE: AM_2901_A
 FAMILY: TTL
 POWER: VCC = P-10, GND = P-30
 DESCRIPTION: BIPOLAR MICROCONTROLLER.
 UNUSED PINS: NONE

FUNCTIONS:

G-CLB = NOT(P-15)
 G-CLBB = NOT(G-CLB)

G-IAB1 = NOT(P-4)
 G-IA1 = INV(G-IAB1)
 G-IBB1 = NOT(P-3)
 G-IB1 = INV(G-IBB1)
 G-ICB1 = NOT(P-2)
 G-IC1 = INV(G-ICB1)
 G-IDB1 = NOT(P-1)
 G-IDD1 = INV(G-IDB1)
 G-AMULT0 = AND(G-IAB1, G-IBB1, G-ICB1, G-IDB1)
 G-AMULT1 = AND(G-IA1, G-IBB1, G-ICB1, G-IDB1)
 G-AMULT2 = AND(G-IAB1, G-IB1, G-ICB1, G-IDB1)
 G-AMULT3 = AND(G-IA1, G-IB1, G-ICB1, G-IDB1)
 G-AMULT4 = AND(G-IAB1, G-IBB1, G-IC1, G-IDB1)
 G-AMULT5 = AND(G-IA1, G-IBB1, G-IC1, G-IDB1)
 G-AMULT6 = AND(G-IAB1, G-IB1, G-IC1, G-IDB1)
 G-AMULT7 = AND(G-IA1, G-IB1, G-IC1, G-IDB1)
 G-AMULT8 = AND(G-IAB1, G-IBB1, G-ICB1, G-IDD1)
 G-AMULT9 = AND(G-IA1, G-IBB1, G-ICB1, G-IDD1)
 G-AMULT10 = AND(G-IAB1, G-IB1, G-ICB1, G-IDD1)
 G-AMULT11 = AND(G-IA1, G-IB1, G-ICB1, G-IDD1)
 G-AMULT12 = AND(G-IAB1, G-IBB1, G-IC1, G-IDD1)
 G-AMULT13 = AND(G-IA1, G-IBB1, G-IC1, G-IDD1)
 G-AMULT14 = AND(G-IAB1, G-IB1, G-IC1, G-IDD1)
 G-AMULT15 = AND(G-IA1, G-IB1, G-IC1, G-IDD1)

G-IAB2 = NOT(P-17)
 G-IA2 = INV(G-IAB2)
 G-IBB2 = NOT(P-18)
 G-IB2 = INV(G-IBB2)
 G-ICB2 = NOT(P-19)
 G-IC2 = INV(G-ICB2)
 G-IDB2 = NOT(P-20)
 G-IDD2 = INV(G-IDB2)
 G-BMULT0 = AND(G-IAB2, G-IBB2, G-ICB2, G-IDB2)
 G-BMULT1 = AND(G-IA2, G-IBB2, G-ICB2, G-IDB2)
 G-BMULT2 = AND(G-IAB2, G-IB2, G-ICB2, G-IDB2)
 G-BMULT3 = AND(G-IA2, G-IB2, G-ICB2, G-IDB2)
 G-BMULT4 = AND(G-IAB2, G-IBB2, G-IC2, G-IDB2)

G-BMULT5 = AND(G-IA2, G-IBB2, G-IC2, G-IDB2)
 G-BMULT6 = AND(G-IAH2, G-IB2, G-IC2, G-IDB2)
 G-BMULT7 = AND(G-IA2, G-IB2, G-IC2, G-IDB2)
 G-BMULT8 = AND(G-IAB2, G-IBB2, G-ICB2, G-IDD2)
 G-BMULT9 = AND(G-IA2, G-IBB2, G-ICB2, G-IDD2)
 G-BMULT10 = AND(G-IAB2, G-IB2, G-ICB2, G-IDD2)
 G-BMULT11 = AND(G-IA2, G-IB2, G-ICB2, G-IDD2)
 G-BMULT12 = AND(G-IAB2, G-IBB2, G-IC2, G-IDD2)
 G-BMULT13 = AND(G-IA2, G-IBB2, G-IC2, G-IDD2)
 G-BMULT14 = AND(G-IAB2, G-IB2, G-IC2, G-IDD2)
 G-BMULT15 = AND(G-IA2, G-IB2, G-IC2, G-IDD2)

G-A00 = AND(FF-RM00); DIS_HIGH(G-AMULT0)
 G-A00 = AND(FF-RM01); DIS_HIGH(G-AMULT1)
 G-A00 = AND(FF-RM02); DIS_HIGH(G-AMULT2)
 G-A00 = AND(FF-RM03); DIS_HIGH(G-AMULT3)
 G-A00 = AND(FF-RM04); DIS_HIGH(G-AMULT4)
 G-A00 = AND(FF-RM05); DIS_HIGH(G-AMULT5)
 G-A00 = AND(FF-RM06); DIS_HIGH(G-AMULT6)
 G-A00 = AND(FF-RM07); DIS_HIGH(G-AMULT7)
 G-A00 = AND(FF-RM08); DIS_HIGH(G-AMULT8)
 G-A00 = AND(FF-RM09); DIS_HIGH(G-AMULT9)
 G-A00 = AND(FF-RM10); DIS_HIGH(G-AMULT10)
 G-A00 = AND(FF-RM11); DIS_HIGH(G-AMULT11)
 G-A00 = AND(FF-RM12); DIS_HIGH(G-AMULT12)
 G-A00 = AND(FF-RM13); DIS_HIGH(G-AMULT13)
 G-A00 = AND(FF-RM14); DIS_HIGH(G-AMULT14)
 G-A00 = AND(FF-RM15); DIS_HIGH(G-AMULT15)
 G-A01 = AND(FF-RM16); DIS_HIGH(G-AMULT0)
 G-A01 = AND(FF-RM17); DIS_HIGH(G-AMULT1)
 G-A01 = AND(FF-RM18); DIS_HIGH(G-AMULT2)
 G-A01 = AND(FF-RM19); DIS_HIGH(G-AMULT3)
 G-A01 = AND(FF-RM20); DIS_HIGH(G-AMULT4)
 G-A01 = AND(FF-RM21); DIS_HIGH(G-AMULT5)
 G-A01 = AND(FF-RM22); DIS_HIGH(G-AMULT6)
 G-A01 = AND(FF-RM23); DIS_HIGH(G-AMULT7)
 G-A01 = AND(FF-RM24); DIS_HIGH(G-AMULT8)
 G-A01 = AND(FF-RM25); DIS_HIGH(G-AMULT9)
 G-A01 = AND(FF-RM26); DIS_HIGH(G-AMULT10)
 G-A01 = AND(FF-RM27); DIS_HIGH(G-AMULT11)
 G-A01 = AND(FF-RM28); DIS_HIGH(G-AMULT12)
 G-A01 = AND(FF-RM29); DIS_HIGH(G-AMULT13)
 G-A01 = AND(FF-RM30); DIS_HIGH(G-AMULT14)
 G-A01 = AND(FF-RM31); DIS_HIGH(G-AMULT15)
 G-A02 = AND(FF-RM32); DIS_HIGH(G-AMULT0)
 G-A02 = AND(FF-RM33); DIS_HIGH(G-AMULT1)
 G-A02 = AND(FF-RM34); DIS_HIGH(G-AMULT2)
 G-A02 = AND(FF-RM35); DIS_HIGH(G-AMULT3)
 G-A02 = AND(FF-RM36); DIS_HIGH(G-AMULT4)
 G-A02 = AND(FF-RM37); DIS_HIGH(G-AMULT5)
 G-A02 = AND(FF-RM38); DIS_HIGH(G-AMULT6)
 G-A02 = AND(FF-RM39); DIS_HIGH(G-AMULT7)
 G-A02 = AND(FF-RM40); DIS_HIGH(G-AMULT8)
 G-A02 = AND(FF-RM41); DIS_HIGH(G-AMULT9)
 G-A02 = AND(FF-RM42); DIS_HIGH(G-AMULT10)
 G-A02 = AND(FF-RM43); DIS_HIGH(G-AMULT11)
 G-A02 = AND(FF-RM44); DIS_HIGH(G-AMULT12)
 G-A02 = AND(FF-RM45); DIS_HIGH(G-AMULT13)
 G-A02 = AND(FF-RM46); DIS_HIGH(G-AMULT14)

G-A02 = AND(FF-RM047); DIS_HIGH(G-AMULT15)
 G-A03 = AND(FF-RM048); DIS_HIGH(G-AMULT0)
 G-A03 = AND(FF-RM049); DIS_HIGH(G-AMULT1)
 G-A03 = AND(FF-RM050); DIS_HIGH(G-AMULT2)
 G-A03 = AND(FF-RM051); DIS_HIGH(G-AMULT3)
 G-A03 = AND(FF-RM052); DIS_HIGH(G-AMULT4)
 G-A03 = AND(FF-RM053); DIS_HIGH(G-AMULT5)
 G-A03 = AND(FF-RM054); DIS_HIGH(G-AMULT6)
 G-A03 = AND(FF-RM055); DIS_HIGH(G-AMULT7)
 G-A03 = AND(FF-RM056); DIS_HIGH(G-AMULT8)
 G-A03 = AND(FF-RM057); DIS_HIGH(G-AMULT9)
 G-A03 = AND(FF-RM058); DIS_HIGH(G-AMULT10)
 G-A03 = AND(FF-RM059); DIS_HIGH(G-AMULT11)
 G-A03 = AND(FF-RM060); DIS_HIGH(G-AMULT12)
 G-A03 = AND(FF-RM061); DIS_HIGH(G-AMULT13)
 G-A03 = AND(FF-RM062); DIS_HIGH(G-AMULT14)
 G-A03 = AND(FF-RM063); DIS_HIGH(G-AMULT15)
 G-B00 = AND(FF-RM00); DIS_HIGH(G-BMULT0)
 G-B00 = AND(FF-RM01); DIS_HIGH(G-BMULT1)
 G-B00 = AND(FF-RM02); DIS_HIGH(G-BMULT2)
 G-B00 = AND(FF-RM03); DIS_HIGH(G-BMULT3)
 G-B00 = AND(FF-RM04); DIS_HIGH(G-BMULT4)
 G-B00 = AND(FF-RM05); DIS_HIGH(G-BMULT5)
 G-B00 = AND(FF-RM06); DIS_HIGH(G-BMULT6)
 G-B00 = AND(FF-RM07); DIS_HIGH(G-BMULT7)
 G-B00 = AND(FF-RM08); DIS_HIGH(G-BMULT8)
 G-B00 = AND(FF-RM09); DIS_HIGH(G-BMULT9)
 G-B00 = AND(FF-RM010); DIS_HIGH(G-BMULT10)
 G-B00 = AND(FF-RM011); DIS_HIGH(G-BMULT11)
 G-B00 = AND(FF-RM012); DIS_HIGH(G-BMULT12)
 G-B00 = AND(FF-RM013); DIS_HIGH(G-BMULT13)
 G-B00 = AND(FF-RM014); DIS_HIGH(G-BMULT14)
 G-B00 = AND(FF-RM015); DIS_HIGH(G-BMULT15)
 G-B01 = AND(FF-RM016); DIS_HIGH(G-BMULT0)
 G-B01 = AND(FF-RM017); DIS_HIGH(G-BMULT1)
 G-B01 = AND(FF-RM018); DIS_HIGH(G-BMULT2)
 G-B01 = AND(FF-RM019); DIS_HIGH(G-BMULT3)
 G-B01 = AND(FF-RM020); DIS_HIGH(G-BMULT4)
 G-B01 = AND(FF-RM021); DIS_HIGH(G-BMULT5)
 G-B01 = AND(FF-RM022); DIS_HIGH(G-BMULT6)
 G-B01 = AND(FF-RM023); DIS_HIGH(G-BMULT7)
 G-B01 = AND(FF-RM024); DIS_HIGH(G-BMULT8)
 G-B01 = AND(FF-RM025); DIS_HIGH(G-BMULT9)
 G-B01 = AND(FF-RM026); DIS_HIGH(G-BMULT10)
 G-B01 = AND(FF-RM027); DIS_HIGH(G-BMULT11)
 G-B01 = AND(FF-RM028); DIS_HIGH(G-BMULT12)
 G-B01 = AND(FF-RM029); DIS_HIGH(G-BMULT13)
 G-B01 = AND(FF-RM030); DIS_HIGH(G-BMULT14)
 G-B01 = AND(FF-RM031); DIS_HIGH(G-BMULT15)
 G-B02 = AND(FF-RM032); DIS_HIGH(G-BMULT0)
 G-B02 = AND(FF-RM033); DIS_HIGH(G-BMULT1)
 G-B02 = AND(FF-RM034); DIS_HIGH(G-BMULT2)
 G-B02 = AND(FF-RM035); DIS_HIGH(G-BMULT3)
 G-B02 = AND(FF-RM036); DIS_HIGH(G-BMULT4)
 G-B02 = AND(FF-RM037); DIS_HIGH(G-BMULT5)
 G-B02 = AND(FF-RM038); DIS_HIGH(G-BMULT6)
 G-B02 = AND(FF-RM039); DIS_HIGH(G-BMULT7)
 G-B02 = AND(FF-RM040); DIS_HIGH(G-BMULT8)
 G-B02 = AND(FF-RM041); DIS_HIGH(G-BMULT9)
 G-B02 = AND(FF-RM042); DIS_HIGH(G-BMULT10)
 G-B02 = AND(FF-RM043); DIS_HIGH(G-BMULT11)

TABLE 31
(CONT'D)

G-B02 = AND(FF-RM044); DIS_HIGH(G-BMULT12)
 G-B02 = AND(FF-RM045); DIS_HIGH(G-BMULT13)
 G-B02 = AND(FF-RM046); DIS_HIGH(G-BMULT14)
 G-B02 = AND(FF-RM047); DIS_HIGH(G-BMULT15)
 G-B03 = AND(FF-RM048); DIS_HIGH(G-BMULT0)
 G-B03 = AND(FF-RM049); DIS_HIGH(G-BMULT1)
 G-B03 = AND(FF-RM050); DIS_HIGH(G-BMULT2)
 G-B03 = AND(FF-RM051); DIS_HIGH(G-BMULT3)
 G-B03 = AND(FF-RM052); DIS_HIGH(G-BMULT4)
 G-B03 = AND(FF-RM053); DIS_HIGH(G-BMULT5)
 G-B03 = AND(FF-RM054); DIS_HIGH(G-BMULT6)
 G-B03 = AND(FF-RM055); DIS_HIGH(G-BMULT7)
 G-B03 = AND(FF-RM056); DIS_HIGH(G-BMULT8)
 G-B03 = AND(FF-RM057); DIS_HIGH(G-BMULT9)
 G-B03 = AND(FF-RM058); DIS_HIGH(G-BMULT10)
 G-B03 = AND(FF-RM059); DIS_HIGH(G-BMULT11)
 G-B03 = AND(FF-RM060); DIS_HIGH(G-BMULT12)
 G-B03 = AND(FF-RM061); DIS_HIGH(G-BMULT13)
 G-B03 = AND(FF-RM062); DIS_HIGH(G-BMULT14)
 G-B03 = AND(FF-RM063); DIS_HIGH(G-BMULT15)

G-LA0 = AND(G-A00, G-CLBB)
 G-LB0 = AND(G-A0L, NOT(G-CLBB))
 G-A0L = OR(G-LA0, G-LB0)
 G-A0B = INV(G-A0L)
 G-LA1 = AND(G-A01, G-CLBB)
 G-LB1 = AND(G-A1L, NOT(G-CLBB))
 G-A1L = OR(G-LA1, G-LB1)
 G-A1B = INV(G-A1L)
 G-LA2 = AND(G-A02, G-CLBB)
 G-LB2 = AND(G-A2L, NOT(G-CLBB))
 G-A2L = OR(G-LA2, G-LB2)
 G-A2B = INV(G-A2L)
 G-LA3 = AND(G-A03, G-CLBB)
 G-LB3 = AND(G-A3L, NOT(G-CLBB))
 G-A3L = OR(G-LA3, G-LB3)
 G-A3B = INV(G-A3L)

G-LA0 = AND(G-B00, G-CLBB)
 G-LB0 = AND(G-B0L, NOT(G-CLBB))
 G-B0L = OR(G-LA0, G-LB0)
 G-LA1 = AND(G-B01, G-CLBB)
 G-LB1 = AND(G-B1L, NOT(G-CLBB))
 G-B1L = OR(G-LA1, G-LB1)
 G-LA2 = AND(G-B02, G-CLBB)
 G-LB2 = AND(G-B2L, NOT(G-CLBB))
 G-B2L = OR(G-LA2, G-LB2)
 G-LA3 = AND(G-B03, G-CLBB)
 G-LB3 = AND(G-B3L, NOT(G-CLBB))
 G-B3L = OR(G-LA3, G-LB3)

G-I0B = NOT(P-12)
 G-I1B = NOT(P-13)
 G-I2B = NOT(P-14)
 G-I2BB = NOT(G-I2B)
 G-ID = NAND(G-I0B, G-I1B)
 G-AROL = AND(G-I2B, G-I1B, G-A0L)
 G-AROR = AND(G-ID, G-I2BB, P-25)

TABLE 31 (CONT'D)

G-ROB = NOR(G-AROL, G-AROR)
 G-ASOL = AND(G-I2BB, G-I1B, G-AOL)
 G-ASOM = AND(INV(G-I0B), G-I2B, G-BOL)
 G-ASON = OR(INV(G-I1B), INV(G-I2BB))
 G-ASOR = AND(G-ASON, G-I0B, FF-Q0)
 G-SOB = NOR(G-ASOL, G-ASOM, G-ASOR)
 G-AR1L = AND(G-I2B, G-I1B, G-A1L)
 G-AR1R = AND(G-ID, G-I2BB, P-24)
 G-R1B = NOR(G-AR1L, G-AR1R)
 G-AS1L = AND(G-I2BB, G-I1B, G-A1L)
 G-AS1M = AND(INV(G-I0B), G-I2B, G-B1L)
 G-AS1N = OR(INV(G-I1B), INV(G-I2BB))
 G-AS1R = AND(G-AS1N, G-I0B, FF-Q1)
 G-S1B = NOR(G-AS1L, G-AS1M, G-AS1R)
 G-AR2L = AND(G-I2B, G-I1B, G-A2L)
 G-AR2R = AND(G-ID, G-I2BB, P-23)
 G-R2B = NOR(G-AR2L, G-AR2R)
 G-AS2L = AND(G-I2BB, G-I1B, G-A2L)
 G-AS2M = AND(INV(G-I0B), G-I2B, G-B2L)
 G-AS2N = OR(INV(G-I1B), INV(G-I2BB))
 G-AS2R = AND(G-AS2N, G-I0B, FF-Q2)
 G-S2B = NOR(G-AS2L, G-AS2M, G-AS2R)
 G-AR3L = AND(G-I2B, G-I1B, G-A3L)
 G-AR3R = AND(G-ID, G-I2BB, P-22)
 G-R3B = NOR(G-AR3L, G-AR3R)
 G-AS3L = AND(G-I2BB, G-I1B, G-A3L)
 G-AS3M = AND(INV(G-I0B), G-I2B, G-B3L)
 G-AS3N = OR(INV(G-I1B), INV(G-I2BB))
 G-AS3R = AND(G-ASON, G-I0B, FF-Q3)
 G-S3B = NOR(G-AS3L, G-AS3M, G-AS3R)

G-CR = NOT(P-26)
 G-CS = NOT(P-28)
 G-FC = NOT(P-27)
 G-INF = AND(G-CS, P-27)
 G-ING = AND(G-FC, P-28, P-26)
 G-INH = NOR(G-INF, G-ING)

G-ROB1 = XNOR(G-CR, G-ROB)
 G-SOB1 = XNOR(G-CS, G-SOB)
 G-P0 = NAND(G-INH, G-ROB1, G-SOB1)
 G-G0 = NOR(G-ROB1, G-SOB1)
 G-R1B1 = XNOR(G-CR, G-R1B)
 G-S1B1 = XNOR(G-CS, G-S1B)
 G-P1 = NAND(G-INH, G-R1B1, G-S1B1)
 G-G1 = NOR(G-R1B1, G-S1B1)
 G-R2B1 = XNOR(G-CR, G-R2B)
 G-S2B1 = XNOR(G-CS, G-S2B)
 G-P2 = NAND(G-INH, G-R2B1, G-S2B1)
 G-G2 = NOR(G-R2B1, G-S2B1)
 G-R3B1 = XNOR(G-CR, G-R3B)
 G-S3B1 = XNOR(G-CS, G-S3B)
 G-P3 = NAND(G-INH, G-R3B1, G-S3B1)
 G-G3 = NOR(G-R3B1, G-S3B1)

G-CN0 = AND(G-INH, P-29)
 G-CAR0 = NOR(G-CN0)
 G-CN1A = AND(G-INH, G-G0)
 G-CN1B = AND(INH, G-P0, P-29)
 G-CAR1 = NOR(CN1A, CN1B)
 G-CN2A = AND(G-INH, G-G1)

TABLE 31 (CONT'D)

G-CN2B = AND(G-P1, G-INH, G-G0)
 G-CN2C = AND(G-INH, G-P1, G-P0, P-29)
 G-CAR2 = NOR(G-CN2A, G-CN2B, G-CN2C)
 G-CN3A = AND(G-INH, G-G2)
 G-CN3B = AND(G-INH, G-P2, G-G1)
 G-CN3C = AND(G-INH, G-P2, G-P1, G-G0)
 G-CN3D = AND(G-INH, G-P2, G-P1, G-P0, P-29)
 G-CAR3 = NOR(G-CN3A, G-CN3B, G-CN3C, G-CN3D)

 G-GB1 = AND(G-G0, G-P1, G-P2, G-P3)
 G-GB2 = AND(G-G1, G-P2, G-P3)
 G-GB3 = AND(G-G2, G-P3)
 G-GB4 = AND(G-G3)
 G-GBAR(P-32) = NOR(G-GB1, G-GB2, G-GB3, G-GB4)
 G-PB1 = NAND(P-29, G-P3, G-P2, G-P1, G-P0)
 G-PBAR(P-35) = NAND(G-P3, G-P2, G-P1, G-P0)
 G-CN4(P-33) = NAND(G-GBAR, G-PB1)
 G-PB3 = AND(G-GBAR, G-PB1)
 G-OVR(P-34) = XOR(G-PB3, G-CAR3)

 G-PG0 = NAND(G-P0, INV(G-G0))
 G-CARA = AND(G-FC, G-CAR0)
 G-F0 = XNOR(G-CARA, G-PG0)
 G-PG1 = NAND(G-P1, INV(G-G1))
 G-CARB = AND(G-FC, G-CAR1)
 G-F1 = XNOR(G-CARB, G-PG1)
 G-PG2 = NAND(G-P2, INV(G-G2))
 G-CARC = AND(G-FC, G-CAR2)
 G-F2 = XNOR(G-CARC, G-PG2)
 G-PG3 = NAND(G-P3, INV(G-G3))
 G-CARD = AND(G-FC, G-CAR3)
 G-F3 = XNOR(G-CARD, G-PG3)
 G-F0B = NOT(G-F0)
 G-F1B = NOT(G-F1)
 G-F2B = NOT(G-F2)
 G-F3B(P-31) = NOT(G-F3)

 G-FEO(P-11) = NOR(G-F0B, G-F1B, G-F2B, G-F3B)

 G-I6B = NOT(P-5)
 G-I7B = NOT(P-6)
 G-NS = NOT(P-7)
 G-I6BB = NOT(G-I6B)
 G-SD = NOR(P-6, G-NS)
 G-SU = NOR(G-NS, G-I7B)
 G-RE = NAND(G-I7B, G-NS)
 G-GT = AND(G-RE, G-NS)
 G-QE = NOR(G-I6BB, G-GT)
 G-SF = NAND(G-I6B, G-RE, G-NS)
 G-WE = AND(G-RE, INV(G-CLBB))
 GTS-Q0(P-21) = NOT(FF-Q0');
 DIS_LOW(G-SU)
 GTS-Q3(P-16) = NOT(FF-Q3');
 DIS_LOW(G-SU)
 GTS-RAM0(P-9) = NOT(G-F0B);
 DIS_LOW(G-SD)
 GTS-RAM3(P-8) = NOT(G-F3B);
 DIS_LOW(G-SU)

 G-SFB = NOT(G-SF)
 G-CE = NOT(P-40)

G-Y0A = AND(G-SF, G-F0)
 G-Y0B = AND(G-SFB, G-A0B)
 G-Y0(P-36) = NOR(G-Y0A, G-Y0B); DIS_LOW(G-CE)
 G-Y1A = AND(G-SF, G-F1)
 G-Y1B = AND(G-SFB, G-A1B)
 G-Y1(P-37) = NOR(G-Y1A, G-Y1B); DIS_LOW(G-CE)
 G-Y2A = AND(G-SF, G-F2)
 G-Y2B = AND(G-SFB, G-A2B)
 G-Y2(P-38) = NOR(G-Y2A, G-Y2B); DIS_LOW(G-CE)
 G-Y3A = AND(G-SF, G-F3)
 G-Y3B = AND(G-SFB, G-A3B)
 G-Y3(P-39) = NOR(G-Y3A, G-Y3B); DIS_LOW(G-CE)

G-QSU0 = AND(G-SU, GTS-Q0)
 G-QNS0 = AND(G-NS, G-F0B)
 G-QSD0 = AND(G-SD, FF-Q1)
 G-QT0 = NOR(G-QSU0, G-QNS0, G-QSD0)
 G-QSU1 = AND(G-SU, FF-Q0)
 G-QNS1 = AND(G-NS, G-F1B)
 G-QSD1 = AND(G-SD, FF-Q2)
 G-QT1 = NOR(G-QSU1, G-QNS1, G-QSD1)
 G-QSU2 = AND(G-SU, FF-Q1)
 G-QNS2 = AND(G-NS, G-F2B)
 G-QSD2 = AND(G-SD, FF-Q3)
 G-QT2 = NOR(G-QSU2, G-QNS2, G-QSD2)
 G-QSU3 = AND(G-SU, FF-Q2)
 G-QNS3 = AND(G-NS, G-F3B)
 G-QSD3 = AND(G-SD, GTS-Q3)
 G-QT3 = NOR(G-QSU3, G-QNS3, G-QSD3)

G-QCLK = AND(INV(G-CLB), G-QE)
 FF-Q0 = D(DATA=G-QT0, CLK_UP=G-QCLK)
 FF-Q1 = D(DATA=G-QT1, CLK_UP=G-QCLK)
 FF-Q2 = D(DATA=G-QT2, CLK_UP=G-QCLK)
 FF-Q3 = D(DATA=G-QT3, CLK_UP=G-QCLK)

G-TSU0 = AND(G-SU, GTS-RAM0)
 G-TNS0 = AND(G-NS, G-F0B)
 G-TSD0 = AND(G-SD, G-F1B)
 G-TT0 = NOR(G-TSU0, G-TNS0, G-TSD0)
 G-TSU1 = AND(G-SU, G-F0B)
 G-TNS1 = AND(G-NS, G-F1B)
 G-TSD1 = AND(G-SD, G-F2B)
 G-TT1 = NOR(G-TSU1, G-TNS1, G-TSD1)
 G-TSU2 = AND(G-SU, G-F1B)
 G-TNS2 = AND(G-NS, G-F2B)
 G-TSD2 = AND(G-SD, G-F3B)
 G-TT2 = NOR(G-TSU2, G-TNS2, G-TSD2)
 G-TSU3 = AND(G-SU, G-F2B)
 G-TNS3 = AND(G-NS, G-F3B)
 G-TSD3 = AND(G-SD, GTS-RAM3)
 G-TT3 = NOR(G-TSU3, G-TNS3, G-TSD3)

G-ATE1 = AND(G-BMULT0, G-WE)
 FF-RM00 = D(DATA=G-TT0, CLK_UP=G-ATE1)
 FF-RM016 = D(DATA=G-TT1, CLK_UP=G-ATE1)
 FF-RM032 = D(DATA=G-TT2, CLK_UP=G-ATE1)
 FF-RM048 = D(DATA=G-TT3, CLK_UP=G-ATE1)
 G-ATE2 = AND(G-BMULT1, G-WE)

```

FF-RM01 = D(DATA=G-TT0,CLK_UP=G-ATE2)
FF-RM017 = D(DATA=G-TT1,CLK_UP=G-ATE2)
FF-RM033 = D(DATA=G-TT2,CLK_UP=G-ATE2)
FF-RM049 = D(DATA=G-TT3,CLK_UP=G-ATE2)
G-ATE3 = AND(G-BMULT2, G-we)
FF-RM02 = D(DATA=G-TT0,CLK_UP=G-ATE3)
FF-RM018 = D(DATA=G-TT1,CLK_UP=G-ATE3)
FF-RM034 = D(DATA=G-TT2,CLK_UP=G-ATE3)
FF-RM050 = D(DATA=G-TT3,CLK_UP=G-ATE3)
G-ATE4 = AND(G-BMULT3, G-we)
FF-RM03 = D(DATA=G-TT0,CLK_UP=G-ATE4)
FF-RM019 = D(DATA=G-TT1,CLK_UP=G-ATE4)
FF-RM035 = D(DATA=G-TT2,CLK_UP=G-ATE4)
FF-RM051 = D(DATA=G-TT3,CLK_UP=G-ATE4)
G-ATE5 = AND(G-BMULT4, G-we)
FF-RM04 = D(DATA=G-TT0,CLK_UP=G-ATE5)
FF-RM020 = D(DATA=G-TT1,CLK_UP=G-ATE5)
FF-RM036 = D(DATA=G-TT2,CLK_UP=G-ATE5)
FF-RM052 = D(DATA=G-TT3,CLK_UP=G-ATE5)
G-ATE6 = AND(G-BMULT5, G-we)
FF-RM05 = D(DATA=G-TT0,CLK_UP=G-ATE6)
FF-RM021 = D(DATA=G-TT1,CLK_UP=G-ATE6)
FF-RM037 = D(DATA=G-TT2,CLK_UP=G-ATE6)
FF-RM053 = D(DATA=G-TT3,CLK_UP=G-ATE6)
G-ATE7 = AND(G-BMULT6, G-we)
FF-RM06 = D(DATA=G-TT0,CLK_UP=G-ATE7)
FF-RM022 = D(DATA=G-TT1,CLK_UP=G-ATE7)
FF-RM038 = D(DATA=G-TT2,CLK_UP=G-ATE7)
FF-RM054 = D(DATA=G-TT3,CLK_UP=G-ATE7)
G-ATE8 = AND(G-BMULT7, G-we)
FF-RM07 = D(DATA=G-TT0,CLK_UP=G-ATE8)
FF-RM023 = D(DATA=G-TT1,CLK_UP=G-ATE8)
FF-RM039 = D(DATA=G-TT2,CLK_UP=G-ATE8)
FF-RM055 = D(DATA=G-TT3,CLK_UP=G-ATE8)
G-ATE9 = AND(G-BMULT8, G-we)
FF-RM08 = D(DATA=G-TT0,CLK_UP=G-ATE9)
FF-RM024 = D(DATA=G-TT1,CLK_UP=G-ATE9)
FF-RM040 = D(DATA=G-TT2,CLK_UP=G-ATE9)
FF-RM056 = D(DATA=G-TT3,CLK_UP=G-ATE9)
G-ATE10 = AND(G-BMULT9, G-we)
FF-RM09 = D(DATA=G-TT0,CLK_UP=G-ATE10)
FF-RM025 = D(DATA=G-TT1,CLK_UP=G-ATE10)
FF-RM041 = D(DATA=G-TT2,CLK_UP=G-ATE10)
FF-RM057 = D(DATA=G-TT3,CLK_UP=G-ATE10)
G-ATE11 = AND(G-BMULT10, G-we)
FF-RM010 = D(DATA=G-TT0,CLK_UP=G-ATE11)
FF-RM026 = D(DATA=G-TT1,CLK_UP=G-ATE11)
FF-RM042 = D(DATA=G-TT2,CLK_UP=G-ATE11)
FF-RM058 = D(DATA=G-TT3,CLK_UP=G-ATE11)
G-ATE12 = AND(G-BMULT11, G-we)
FF-RM011 = D(DATA=G-TT0,CLK_UP=G-ATE12)
FF-RM027 = D(DATA=G-TT1,CLK_UP=G-ATE12)
FF-RM043 = D(DATA=G-TT2,CLK_UP=G-ATE12)
FF-RM059 = D(DATA=G-TT3,CLK_UP=G-ATE12)
G-ATE13 = AND(G-BMULT12, G-we)
FF-RM012 = D(DATA=G-TT0,CLK_UP=G-ATE13)
FF-RM028 = D(DATA=G-TT1,CLK_UP=G-ATE13)
FF-RM044 = D(DATA=G-TT2,CLK_UP=G-ATE13)
FF-RM060 = D(DATA=G-TT3,CLK_UP=G-ATE13)
G-ATE14 = AND(G-BMULT13, G-we)
FF-RM013 = D(DATA=G-TT0,CLK_UP=G-ATE14)

```

TABLE 31
(CONT'D)

```

FF-RM029 = D(DATA=G-TT1,CLK_UP=G-ATE14)
FF-RM045 = D(DATA=G-TT2,CLK_UP=G-ATE14)
FF-RM061 = D(DATA=G-TT3,CLK_UP=G-ATE14)
G-ATE15 = AND(G-BMULT14, G-WE)
FF-RM014 = D(DATA=G-TT0,CLK_UP=G-ATE15)
FF-RM030 = D(DATA=G-TT1,CLK_UP=G-ATE15)
FF-RM046 = D(DATA=G-TT2,CLK_UP=G-ATE15)
FF-RM062 = D(DATA=G-TT3,CLK_UP=G-ATE15)
G-ATE16 = AND(G-BMULT15, G-WE)
FF-RM015 = D(DATA=G-TT0,CLK_UP=G-ATE16)
FF-RM031 = D(DATA=G-TT1,CLK_UP=G-ATE16)
FF-RM047 = D(DATA=G-TT2,CLK_UP=G-ATE16)
FF-RM063 = D(DATA=G-TT3,CLK_UP=G-ATE16)

```

\$ END CHIP \$

TABLE 31 (CONT'D)

\$ CHIP DEFINITION \$

TYPE: AM_2902
 FAMILY: TTL
 POWER: VCC = P-16, GND = P-8
 DESCRIPTION: LOOK-AHEAD CARRY GENERATORS.
 UNUSED PINS: NONE

FUNCTIONS:

G-CN' = NOT(P-13)
 G-CX0 = AND(P-3, G-CN')
 G-CX1 = AND(P-4, P-3)
 G-CNX(P-12) = NOR(G-CX0, G-CX1)
 G-CY0 = AND(P-2, P-1)
 G-CY1 = AND(P-4, P-1, P-3)
 G-CY2 = AND(P-1, P-3, G-CN')
 G-CNY(P-11) = NOR(G-CY0, G-CY1, G-CY2)
 G-Z3 = AND(P-15, P-14)
 G-Z2 = AND(P-2, P-14, P-1)
 G-Z1 = AND(P-4, P-14, P-1, P-3)
 G-Z0 = AND(P-14, P-1, P-3, G-CN')
 G-CNZ(P-9) = NOR(G-Z0, G-Z1, G-Z2, G-Z3)
 G-Y3 = AND(P-6, P-5)
 G-Y2 = AND(P-15, P-5, P-14)
 G-Y1 = AND(P-2, P-5, P-14, P-1)
 G-Y0 = AND(P-5, P-14, P-1, P-3)
 G-Y(P-10) = OR(G-Y0, G-Y1, G-Y2, G-Y3)
 G-X(P-7) = OR(P-6, P-15, P-2, P-4)

\$ END CHIP \$

APPENDIX B

CPU CARD CHIP LIBRARY IN BLISS


```

MODULE TCS113(language(%bliss36(bliss36) %bliss32(bliss32)),
  addressing_mode(external = long_relative,
    nonexternal = long_relative) ) =
BEGIN  %( module tcs )%

require 'RTNEST,R32';

```

```

GLOBAL ROUTINE IC113(CL1,K1,J1,P1,Q1,Q1B,GND,Q2B,Q2,
  P2,J2,K2,CL2,J) : novalue =

```

```

BEGIN  %( routine tc113 )%

```

```

  OWN    S,R;
  SWITCHES NOOPTIMIZE;

```

```

!           S  CHIP  DEFINITION  S
!
!   TYPE:           54_LS_113
!
!   FAMILY:         TTL
!
!   POWER:          VCC = P-14,  GND = P-7
!
!   DESCRIPTION:    DUAL GATED J - K FLIP-FLOP.
!
!   UNUSED PINS:    NONE
!
!
!

```

```

!   FUNCTIONS:
!

```

```

S = -1; FF32(S,S,S,.Q1,.Q1B);
NAND4(S,.P1,.Q1B,.J1,.CL1);
NAND3(R,.Q1,.K1,.CL1);
FF32(.P1,S,R,.Q1,.Q1B);

```

```

!   FF-A(P-5, P-6) = J-K (PRESET=INV(P-4)
!                                     J=P-3
!                                     K=P-2
!                                     CLK_UP=P-1)
!

```

```

S = -1; FF32(S,S,S,.Q2,.Q2B);
NAND4(S,.P2,.Q2B,.J2,.CL2);
NAND3(R,.Q2,.K2,.CL2);
FF32(.P2,S,R,.Q2,.Q2B);

```

```

!   FF-B(P-9, P-8) = J-K (PRESET=INV(P-10)

```

TABLE 33

```

J=P-11
K=P-12
CLK_UP=P-13
$ END CHIP $

```

```
END % ( routine tc113 ) %;
```

```

:      NOTE:  SOURCE DATA - "THE TTL DATA BOOK
:              FOR DESIGN ENGINEERS",
:              SECOND EDITION

```

END ELUDOM

TABLE 33 (CONT'D)


```

!      G-02 = AND(G-STB, P-2, G-IAB, G-IB, G-ICB)
      AND5(O[3],.I3,STB,IA,IB,ICB);
!      G-03 = AND(G-STB, P-1, G-IA, G-IB, G-ICB)
      AND5(O[4],.I4,STB,IAB,IBB,IC);
!      G-04 = AND(G-STB, P-15, G-IAB, G-IBB, G-IC)
      AND5(O[5],.I5,STB,IA,IBB,IC);
!      G-05 = AND(G-STB, P-14, G-IA, G-IBB, G-IC)
      AND5(O[6],.I6,STB,IAB,IB,IC);
!      G-06 = AND(G-STB, P-13, G-IAB, G-IB, G-IC)
      AND5(O[7],.I7,STB,IA,IB,IC);
!      G-07 = AND(G-STB, P-12, G-IA, G-IB, G-IC)
!
!      OR8(STB,U);
!      .W = NOT .STB;
!      G-W(P-6) = NOR(G-00, G-01, G-02, G-03, G-04,
!                  G-05, G-06, G-07)
!
!      INV(.Y,.w);
!      G-Y(P-5) = INV(G-W)
!
!                  $ END CHIP $

END  %( routine tC151 )% ;

      !      NOTE:  SOURCE DATA - "THE TTL DATA BOOK
      !                  FOR DESIGN ENGINEERS",
      !                  SECOND EDITION
END  ELUDOM  %( MODULE  TCS151 )%

```

TABLE 34 (CONT'D)

```

MODULE TC151(language(%bliss36(bliss36) %bliss32(bliss32)),
addressing_mode(external = long_relative,
nonexternal = long_relative) ) =
BEGIN  %( module tcs )%

require 'RTNFST.R32';

```

```

GLOBAL ROUTINE TC151(I3,I2,I1,I0,Y,W,STROBE,GND,C,B,
A,I7,I6,I5,I4,J) : novalue =
BEGIN  %( routine tc151 )%

```

```

LOCAL O : vector[8],
STB,IA,IAB,IB,IBB,IC,ICB;

```

```

!           $ CHIP DEFINITION $
!
!   TYPE:           54_LS_151
!
!   FAMILY:         TTL
!
!   POWER:          VCC = P-16, GND = P-8
!
!   DESCRIPTION:    1 - OF - 8 DATA SELECTORS/
!                   MULTIPLEXERS.
!
!   UNUSED PINS:    NONE
!
!   FUNCTIONS:
!
!   INV(IAB,.A); INV(IA,IAB);
!   G-IAB = NOT(P-11)
!   G-IA = INV(G-IAB)
!
!   INV(IBB,.B); INV(IB,IBB);
!   G-IBB = NOT(P-10)
!   G-IB = INV(G-IBB)
!
!   INV(ICB,.C); INV(IC,ICB);
!   G-ICB = NOT(P-9)
!   G-IC = INV(G-ICB)
!
!   INV(STB,.STROBE);
!   G-STB = INV(P-7)
!
!   AND5(O[0],.I0,STB,IAB,IBB,ICB);
!   G-O0 = AND(G-STB, P-4, G-IAB, G-IBB, G-ICB)
!
!   AND5(O[1],.I1,STB,IA,IBB,ICB);
!   G-O1 = AND(G-STB, P-3, G-IA, G-IBB, G-ICB)
!
!   AND5(O[2],.I2,STB,IA,IB,ICB);

```

```

!      G-02 = AND(G-STB, P-2, G-IAB, G-IB, G-ICB)
      AND5(O[3],.I3,STB,IA,IB,ICB);
!      G-03 = AND(G-STB, P-1, G-IA, G-IB, G-ICB)
      AND5(O[4],.I4,STB,IAB,IBB,IC);
!      G-04 = AND(G-STB, P-15, G-IAB, G-IBB, G-IC)
      AND5(O[5],.I5,STB,IA,IBB,IC);
!      G-05 = AND(G-STB, P-14, G-IA, G-IBB, G-IC)
      AND5(O[6],.I6,STB,IAB,IB,IC);
!      G-06 = AND(G-STB, P-13, G-IAB, G-IB, G-IC)
      AND5(O[7],.I7,STB,IA,IB,IC);
!      G-07 = AND(G-STB, P-12, G-IA, G-IB, G-IC)
!
!      OR8(STB,0);
      .W = NOT ,STB;
!      G-W(P-6) = NOR(G-00, G-01, G-02, G-03, G-04,
!                  G-05, G-06, G-07)
!
!      INV(.Y,.w);
!      G-Y(P-5) = INV(G-w)
!
!      S END CHIP S

END  %( routine tc151 )% ;

      !      NOTE:  SOURCE DATA - "THE TTL DATA BOOK
      !      !      FOR DESIGN ENGINEERS",
      !      !      SECOND EDITION
END  ELUDOM  %( MODULE  TCS151 )%

```

TABLE 35 (CONT'D)

```

MODULE TCS153(language(%bliss36(bliss36) %bliss32(bliss32)),
addressing_mode(external = long_relative,
nonexternal = long_relative) ) =
BEGIN  %( module tcs )%

require 'RTNEST,R32';

MACRO   ORR4(R,I1,I2,I3,I4,J) =
        R=(.I1 OR .I2 OR .I3 OR .I4) FAND(J,0)
        FORR(J,1)%;

GLOBAL ROUTINE TC153(G1,B,I13,I12,I11,I10,R1,GND,R2,I20,
I21,I22,I23,A,G2,J) : novalue =
BEGIN  %( routine tc153 )%

        LOCAL SBAR,IA,IAB,IB,IBB,O1,O2,O3,O4;

!
!           $ CHIP DEFINITION $
!
!   TYPE:           54_LS_153
!
!   FAMILY:         TTL
!
!   POWER:          VCC = P-16, GND = P-8
!
!   DESCRIPTION:    DUAL 1 OF 4 DATA SELECTORS/MULTIPLEXERS.
!
!   UNUSED PINS:    NONE
!
!
!           FUNCTIONS:
!
!
!   INV(SBAR,.G1); INV(IAB,.A); INV(IA,IAB);
!
!           G-SBARA = INV(P-1)
!           G-IAB = NOT(P-14)
!           G-IA = INV(G-IAB)
!
!   INV(IBB,.B); INV(IB,IBB);
!           G-IBB = NOT(P-2)
!           G-IB = INV(G-IBB)
!
!   AND4(G1,SBAR,IAB,IBB,.I10);
!           G-O1A = AND(G-SBARA, P-6, G-IAB, G-IBB)
!
!   AND4(O2,SBAR,IA,IBB,.I11);
!           G-O2A = AND(G-SBARA, P-5, G-IA, G-IBB)
!
!   AND4(O3,SBAR,IAB,IB,.I12);
!           G-O3A = AND(G-SBARA, P-4, G-IAB, G-IB)
!
!   AND4(O4,SBAR,IA,IB,.I13);
!           G-O4A = AND(G-SBARA, P-3, G-IA, G-IB)
!
!   ORR4(.R1,G1,O2,O3,O4);

```

```

!           G-R1(P-7) = OR(G-U1A, G-U2A, G-U3A, G-U4A)
!
      INV(SBAR,,G2);
!           G-SBARB = INV(P-15)
      AND4(O1,SBAR,IAB,IBB,,I20);
!           G-O1B = AND(G-SBARB, P-10, G-IAB, G-IBB)
      AND4(O2,SBAR,IA,IBB,,I21);
!           G-O2B = AND(G-SBARB, P-11, G-IA, G-IBB)
      AND4(O3,SBAR,IAB,IB,,I22);
!           G-O3B = AND(G-SBARB, P-12, G-IAB, G-IB)
      AND4(O4,SBAR,IA,IB,,I23);
!           G-O4B = AND(G-SBARB, P-13, G-IA, G-IB)

      ORR4(.R2,O1,O2,O3,O4);
!           G-R2(P-9) = OR(G-U1B, G-O2B, G-O3B, G-O4B)
!

!           $ END CHIP $

END      %( ROUTINE TC153 )% ;

!           NOTE:  SOURCE DATA - "THE TTL DATA BOOK
!                   FOR DESIGN ENGINEERS",
!                   SECOND EDITION

END      ELUDOM

```

TABLE 36 (CONT'D)


```

MODULE TCS158(language(%bliss36(bliss36) %bliss32(bliss32)),
addressing_mode(external = long_relative,
nonexternal = long_relative) ) =
BEGIN %( module tcs )%

require 'RTNEST.R32';

```

```

GLOBAL ROUTINE TC158(S,A1,B1,Y1,A2,B2,Y2,GND,Y3,B3,A3,
Y4,B4,A4,G,J) : novalue =
BEGIN %( routine tc158 )%

```

```

LOCAL A,B,C,D;

```

```

!           S  CHIP  DEFINITION  S
!
!
!   TYPE:           54_LS_158
!   FAMILY:         TTL
!   POWER:          VCC = P-16, GND = P-8
!   DESCRIPTION:    QUAD 2-LINE TO 1-LINE DATA
!                   SELECTORS/MULTIPLEXERS.
!   UNUSED PINS:    NONE
!
!   FUNCTIONS:
!
!       INV(A,,S);
!           G-SB = NOT(P-1)
!
!       NOR(B,A,.G);
!           G-SY = AND(INV(G-SB), INV(P-15))
!
!       NOR(A,,S,.G);
!           G-SX = AND(INV(P-1), INV(P-15))
!
!       ANDR(C,.A1,A);
!           G-A1 = AND(P-2, G-SX)
!
!       ANDR(D,.B1,B);
!           G-B1 = AND(P-3, G-SY)
!
!       NOR(.Y1,C,D);
!           G-Q1(P-4) = NOR(G-A1, G-B1)
!
!       ANDR(C,.A2,A);
!           G-A2 = AND(P-5, G-SX)

```

```

      ANDR(D,.B2,B);
!      G-B2 = AND(P-6, G-SY)

      NOR(.Y2,C,D);
!      G-02(P-7) = NOR(G-A2, G-B2)
!

      ANDR(C,.A3,A);
!      G-A3 = AND(P-11, G-SX)

      ANDR(D,.B3,B);
!      G-B3 = AND(P-10, G-SY)

      NOR(.Y3,C,D);
!      G-03(P-9) = NOR(G-A3, G-B3)
!

      ANDR(C,.A4,A);
!      G-A4 = AND(P-14, G-SX)

      ANDR(D,.B4,B);
!      G-B4 = AND(P-13, G-SY)

      NOR(.Y4,C,D);
!      G-04(P-12) = NOR(G-A4, G-B4)
!

      END  %( ROUTINE TC158 )% ;

!      $ END CHIP $

```

```

!      NOTE:  SOURCE DATA - "THE TTL DATA BOOK
!                                     FOR DESIGN ENGINEERS",
!                                     SECOND EDITION
END
ELUDOM

```

TABLE 37 (CONT'D)

```

MODULE TC169(language(%bliss36(bliss36) %bliss32(bliss32)),
addressing_mode(external = long_relative,
nonexternal = long_relative) ) =
BEGIN  %( module tcs )%

```

```

require 'RTNEST.R32';

```

```

GLOBAL ROUTINE TC169(UD,CK,IA,IB,IC,ID,EPBAR,K,LOAD,ETBAR,
QD,QC,QB,QA,RCQ,J) : novalue =
BEGIN  %( routine tc169 )%

```

```

OWN  CLOCK,UDI,LB,L,EP,ET,COUNT,C1,C2,C3,C4,I1,I2,I3,I4;
OWN  A,B,D1,D2,D3,D4;

```

```

!      $ CHIP DEFINITION $

```

```

!      TYPE:          54_LS_169

```

```

!      FAMILY:        TTL

```

```

!      POWER:         VCC = P-16, GND = P-8

```

```

!      DESCRIPTION:    SYNCHRONOUS 4-BIT UP/DOWN COUNTER.

```

```

!      UNUSED PINS:    NONE

```

```

!      FUNCTIONS:

```

```

!      BUF(CLOCK,,CK); INV(UDI,,UD);
!      G-CK = NOT(P-2)
!      G-UDI = NOT(P-1)

```

```

!      INV(LB,,LOAD);
!      G-LB = INV(P-9)

```

```

!      INV(L,LB); INV(EP,,EPBAR);
!      G-L = NOT(G-LB)
!      G-EP = INV(P-7)

```

```

!      INV(ET,,ETBAR);
!      G-ET = INV(P-10)
!      COUNT = AND3(.LOAD,EP,ET);
!      G-COUNT = AND(P-9, G-EP, G-ET)

```

```

!      A = NOT ,UDI;
!      C1 = (NOT(ANDF(UDI, ST[,K]) OR ANDF(ST[,K+1], A)));

```

```

!      G-QA1 = AND(G-UDI, FF-A)
!      G-QA2 = AND(INV(G-UDI), FF-A')

```

G-C1 = NOR(G-QA1, G-QA2)

A = NOT .UDI;
C2 = (NOT(ANDF(UDI, ST[.K+2]) OR ANDF(ST[.K+3], A)));

G-QB1 = AND(G-UDI, FF-B)
G-QB2 = AND(INV(G-UDI), FF-B')
G-C2 = NOR(G-QB1, G-QB2)

A = NOT .UDI;
C3 = (NOT(ANDF(UDI, ST[.K+4]) OR ANDF(ST[.K+5], A)));

G-QC1 = AND(G-UDI, FF-C)
G-QC2 = AND(INV(G-UDI), FF-C')
G-C3 = NOR(G-QC1, G-QC2)

A = NOT .UDI;
C4 = (NOT(ANDF(UDI, ST[.K+6]) OR ANDF(ST[.K+7], A)));

G-QD1 = AND(G-UDI, FF-DX)
G-QD2 = AND(INV(G-UDI), FF-DX')
G-C4 = NOR(G-QD1, G-QD2)

INV(A,COUNT);
ANDR(B,COUNT,ST[.K+1]);

G-A = NOT(G-COUNT)
G-AB = AND(G-COUNT, FF-A')

D1 = (AND3(ST[.K], A, L) OR .B OR ANDF(.IA,LB));

G-AA = AND(FF-A, G-A, G-L)
G-IA = AND(G-LB, P-3)
G-D1 = OR(G-AA, G-AB, G-IA)

NAND(A,COUNT,C1);
B = AND3(C1,COUNT,ST[.K+3]);

G-B = NAND(G-COUNT, G-C1)
G-BB = AND(G-COUNT, FF-B', G-C1)

D2 = (AND3(ST[.K+2], A, L) OR .B OR ANDF(.IB,LB));

G-BA = AND(FF-B, G-B, G-L)
G-IB = AND(G-LB, P-4)
G-D2 = OR(G-BA, G-BB, G-IB)

NAND3(A,COUNT,C1,C2);
AND4(B,COUNT,C2,C1,ST[.K+5]);

G-C = NAND(G-COUNT, G-C1, G-C2)
G-CB = AND(G-COUNT, FF-C' G-C1, G-C2)

D3 = (AND3(ST[K+4], A, L) OR .B OR ANDF(.IC,LB));

G-CA = AND(FF-C, G-C, G-L)
G-IC = AND(G-LB, P-5)
G-D3 = OR(G-CA, G-CB, G-IC)

NAND4(A,COUNT,C1,C2,C3);
AND5(B,COUNT,C3,C2,C1,ST[K+7]);

G-DX = NAND(G-COUNT, G-C1, G-C2, G-C3)
G-DB = AND(G-COUNT, FF-DX', G-C1, G-C2, G-C3)

D4 = (AND3(ST[K+6], A, L) OR .B OR ANDF(.ID,LB));

G-DA = AND(FF-DX, G-DX, G-L)
G-ID = AND(G-LB, P-6)
G-D4 = OR(G-DA, G-DB, G-ID)

NAND5(.RCO,ET,C1,C2,C3,C4);
G-CO(P-15) = NAND(INV(P-10), G-C1, G-C2, G-C3, G-C4)

A = (.D1 AND .CLOCK) OR (.ST[K] AND NOT .CLOCK);
BUF(ST[K], A);
INV((ST[K+1]), A);

FF-A(P-14) = D(CLK_UP=G-CK,
DATA=G-D1)

A = (.D2 AND .CLOCK) OR (.ST[K+2] AND NOT .CLOCK);
BUF(ST[K+2], A);
INV((ST[K+3]), A);

FF-B(P-13) = D(CLK_UP=G-CK,
DATA=G-D2)

A = (.D3 AND .CLOCK) OR (.ST[K+4] AND NOT .CLOCK);
BUF(ST[K+4], A);
INV((ST[K+5]), A);

FF-C(P-12) = D(CLK_UP=G-CK,
DATA=G-D3)

A = (.D4 AND .CLOCK) OR (.ST[K+6] AND NOT .CLOCK);
BUF(ST[K+6], A);
INV((ST[K+7]), A);

FF-DX(P-11) = D(CLK_UP=G-CK,
DATA=G-D4)

\$ END CHIP \$

.QA = .ST[.K]; .QB = .ST[.K+2]; .QC = .ST[.K+4];
.QD = .ST[.K+6];

END % (routine tc169) %;

! NOTE: SOURCE DATA - "THE TTL DATA BOOK
! FOR DESIGN ENGINEERS",
! SECOND EDITION --
! RE-DRAWN FOR CLARITY

END ELUDOM

TABLE 38 (CONT'D)

```

MODULE TC175(language(%bliss36(bliss36) %bliss32(bliss32)),
  addressing_mode(external = long_relative,
    nonexternal = long_relative) ) =
BEGIN  %( module tcs )%

require 'RTNEST,R32';

```

```

GLOBAL ROUTINE TC175(CLEAR,Q1,Q1B,D1,D2,Q2B,Q2,K,
  CLOCK,Q3,Q3B,D3,D4,Q4B,Q4,J) : novalue =
BEGIN  %( routine tc175 )%

```

```

  OWN  CLB,CLR,A,B;

```

```

!           $ CHIP DEFINITION $
!
!
!   TYPE:           54_LS_175
!
!   FAMILY:         TTL
!
!   POWER:          VCC = P-16, GND = P-8
!
!   DESCRIPTION:    QUAD D-TYPE FLIP-FLOPS
!
!   UNUSED PINS:    NONE
!
!   FUNCTIONS:
!
!   BUF(CLB,.CLOCK);
!     G-CLK=NOT(P-9)
!
!   INV(CLR,.CLEAR);
!     G-CLR=NOT(INV(P-1))
!
!
!   ST[.K] = (..D1 AND .CLB) OR (.ST[.K] AND NOT .CLB);
!   ST[.K] = .ST[.K] AND NOT .CLR;
!   .Q1 = .ST[.K];
!   .Q1B = NOT .ST[.K];
!
!   FF-Q1(P-2,P-3) = D(CLEAR=INV(G-CLR),
!                       DATA=P-4,
!                       CLK_UP=INV(G-CLK))
!
!   ST[.K+2] = (..D2 AND .CLB) OR (.ST[.K+2] AND NOT .CLB);
!   ST[.K+2] = .ST[.K+2] AND NOT .CLR;
!   .Q2 = .ST[.K+2];
!   .Q2B = NOT .ST[.K+2];
!
!   FF-Q2(P-7,P-6) = D(CLEAR=INV(G-CLR),
!                       DATA=P-5,

```



```

MODULE TC253(language(%bliss36(bliss36) %bliss32(bliss32)),
  addressing_mode(external = long_relative,
    nonexternal = long_relative) ) =
BEGIN  %( module tcs )%

require "RTNEST,RJ2";

MACRO  ORR4(R,I1,I2,I3,I4,J) =
      R=(.I1 OR .I2 OR .I3 OR .I4) FAND(J,0)
      FORR(J,1)%;

GLOBAL ROUTINE TC253(G1,B,I13,I12,I11,I10,R1,GND,R2,I20,
  I21,I22,I23,A,G2,J) : novalue =
BEGIN  %( routine tc253 )%

  LOCAL SBAR,IA,IAB,IB,IBB,U1,U2,U3,U4,OUT;

```

```

!           S  CHIP  DEFINITION  S
!
!
!   TYPE:           54_LS_253
!
!   FAMILY:         TTL
!
!   POWER:          VCC = P-16,  GND = P-8
!
!   DESCRIPTION:    DUAL 4-LINE-TO-1-LINE DATA
!                   SELECTORS/MULTIPLEXERS
!                   WITH 3-STATE OUTPUTS.
!
!   UNUSED PINS:    NONE
!
!   FUNCTIONS:
!
!   INV(SBAR,.G1); INV(IAB,.A); INV(IA,IAB);
!   G-SBAR1 = INV(P-1)
!   G-IAB = NOT(P-14)
!   G-IA = INV(G-IAB)
!
!   INV(IBB,.B); INV(IB,IBB);
!   G-IBB = NOT(P-2)
!   G-IB = INV(G-IBB)
!
!   AND4(U1,SBAR,IAB,IBB,.I10);
!   G-U1A = AND(G-SBAR1, P-6, G-IAB, G-IBB)

```

```

AND4(O2,SBAR,IA,IBB,.I11);
!      G-02A = AND(G-SBAR1, P-5, G-IA, G-IBB)

AND4(O3,SBAR,IAB,IB,.I12);
!      G-03A = AND(G-SBAR1, P-4, G-IAB, G-IB)

AND4(O4,SBAR,IA,IB,.I13);
!      G-04A = AND(G-SBAR1, P-3, G-IA, G-IB)

ORR4(OUT,O1,O2,O3,O4);
.R1= (.OUT OR NOT .SBAR);
!      GTS-1Y(P-7) = OR(G-01A, G-02A, G-03A, G-04A);  DIS_LOW(G-SBAR1
!

INV(SBAR,.G2);
!      G-SBAR2 = INV(P-15)
!
AND4(O1,SBAR,IAB,IBB,.I20);
!      G-01B = AND(G-SBAR2, P-10, G-IAB, G-IBB)

AND4(O2,SBAR,IA,IBB,.I21);
!      G-02B = AND(G-SBAR2, P-11, G-IA, G-IBB)

AND4(O3,SBAR,IAB,IB,.I22);
!      G-03B = AND(G-SBAR2, P-12, G-IAB, G-IB)

AND4(O4,SBAR,IA,IB,.I23);
!      G-04B = AND(G-SBAR2, P-14, G-IA, G-IB)

ORR4(OUT,O1,O2,O3,O4);
.R2= (.OUT OR NOT .SBAR);
!      GTS-2Y(P-9) = OR(G-01B, G-02B, G-03B, G-04B);  DIS_LOW(G-SBAR2
!
!
!      $ END CHIP $

```

```

!      NOTE:  SOURCE DATA - "THE TTL DATA BOOK
!                                     FOR DESIGN ENGINEERS",
!                                     SECOND EDITION AND
!                                     LOGIC DIAGRAM ATTACHED.
!

```

END %(routine tc253)% ;

END

ELUDOM

```
MODULE TC273(language(%bliss36(bliss36) %bliss32(bliss32)),
addressing_mode(external = long_relative,
nonexternal = long_relative) ) =
BEGIN %( module tcs )%

require 'RTNEST.R32';
```

```
GLOBAL ROUTINE TC273(CLEAR,Q1,D1,D2,Q2,Q3,D3,D4,Q4,K,  
                    CLOCK,Q5,D5,D6,Q6,Q7,D7,D8,Q8,J) : novalue =  
BEGIN    %( routine tc273 )%
```

LOCAL CLB,CLR,A,B;

! \$ CHIP DEFINITION \$

TYPE:	54 _{LS} 273
FAMILY:	TTL
POWER:	VCC = P-20, G-ND = P-10
DESCRIPTION:	OCTAL D-TYPE FLIP-FLOPS.
UNUSED PINS:	NONE

! FUNCTIONS:

```
BUF(CLB,,CLOCK); INV(CLR,,CLEAR);
```

```

!      G-CLB  =  NOT(P-11)
!      G-CLR  =  NOT(INV(P-1))
!

```

```

      A = (.,D1 AND .CLB) OR (.,ST[.K] AND NOT .CLB);
      INV (B, CLR);      A = .A AND .B;
      BUF (ST[.K], A);
      INV ((ST[.K]+%UPVAL), A);
      BUF(.,Q1,ST[.K]);

```

```

!      FF-D1(P-2)  =  D(CLEAR=INV(G-CLR),
!                      DATA=P-3,
!                      CLK_UP=INV(G-CLB))

```

```
A = (...D2 AND ,CLB) OR (,ST[,K+2] AND NOT ,CLB);
```

```

    INV (B, CLR);    A = .A AND .B;
    BUF (ST[.K+2], A);
    INV ((ST[.K+2]+%UPVAL), A);
    BUF(.Q2,ST[.K+2]);

```

```

!      FF-D2(P=5)  =  D(CLEAR=INV(G=CLR),
!                      DATA=P=4,
!                      CLK_UP=INV(G=CLB))
!

```

```

    A = (.,D3 AND .CLB) OR (.ST[.K+4] AND NOT .CLB);
    INV (B, CLR);    A = .A AND .B;
    BUF (ST[.K+4], A);
    INV ((ST[.K+4]+%UPVAL), A);
    BUF(.Q3,ST[.K+4]);

```

```

!      FF-D3(P=6)  =  D(CLEAR=INV(G=CLR),
!                      DATA=P=7,
!                      CLK_UP=INV(G=CLB))
!

```

```

    A = (.,D4 AND .CLB) OR (.ST[.K+6] AND NOT .CLB);
    INV (B, CLR);    A = .A AND .B;
    BUF (ST[.K+6], A);
    INV ((ST[.K+6]+%UPVAL), A);
    BUF(.Q4,ST[.K+6]);

```

```

!      FF-D4(P=9)  =  D(CLEAR=INV(G=CLR),
!                      DATA=P=8,
!                      CLK_UP=INV(G=CLB))
!

```

```

    A = (.,D5 AND .CLB) OR (.ST[.K+8] AND NOT .CLB);
    INV (B, CLR);    A = .A AND .B;
    BUF (ST[.K+8], A);
    INV ((ST[.K+8]+%UPVAL), A);
    BUF(.Q5,ST[.K+8]);

```

```

!      FF-D5(P=12) =  D(CLEAR=INV(G=CLR,
!                      DATA=P=13,
!                      CLK_UP=INV(G=CLB))
!

```

```

    A = (.,D6 AND .CLB) OR (.ST[.K+10] AND NOT .CLB);
    INV (B, CLR);    A = .A AND .B;
    BUF (ST[.K+10], A);
    INV ((ST[.K+10]+%UPVAL), A);
    BUF(.Q6,ST[.K+10]);

```

```

!      FF-D6(P=15) =  D(CLEAR=INV(G=CLR,
!                      DATA=P=14,
!                      CLK_UP=INV(G=CLB))
!

```

```

    A = (.,D7 AND .CLB) OR (.ST[.K+12] AND NOT .CLB);
    INV (B, CLR);    A = .A AND .B;
    BUF (ST[.K+12], A);
    INV ((ST[.K+12]+%UPVAL), A);
    BUF(.Q7,ST[.K+12]);

```

```

!      FF-D7(P=16) =  D(CLEAR=INV(G=CLR,
!                      DATA=P=17,
!                      CLK_UP=INV(G=CLB))
!

```

```

    A = (.,D8 AND .CLB) OR (.ST[.K+14] AND NOT .CLB);

```

```

      INV (B, CLR);  A = .A AND .B;
      BUF (ST[,K+14], A);
      INV ((ST[,K+14]+%UPVAL), A);
      BUF(,QB,ST[,K+14]);

!      FF=D8(P=19) = D(CLEAR=INV(G=CLR,
!                      DATA=P=18,
!                      CLK_UP=INV(G=CLB))
!

```

```

END %( routine tc273 )%;

```

```

!          $ END CHIP $

```

```

!      NOTE:  SOURCE DATA - "THE TTL DATA BOOK
!                      FOR DESIGN ENGINEERS",
!                      SECOND EDITION

```

```

END

```

```

ELUDOM

```

```

MODULE TC352(language(%bliss36(bliss36) %bliss32(bliss32)),
addressing_mode(external = long_relative,
nonexternal = long_relative) ) =
BEGIN %( module tcs )%

```

```

require 'RTNEST.R32';

```

```

MACRO ORR4(R,I1,I2,I3,I4,J) =
R=(.I1 OR .I2 OR .I3 OR .I4) FAND(J,0)
FORR(J,1)%;

```

```

GLOBAL ROUTINE TC352(G1,B,I13,I12,I11,I10,R1,GND,R2,I20,
I21,I22,I23,A,G2,J) : novalue =
BEGIN %( routine tc352 )%

```

```

LOCAL SBAR,IA,IAB,IB,IBB,O1,O2,O3,O4,OUT;

```

```

!           $ CHIP DEFINITION $
!

```

```

!      TYPE:           54_LS_352
!
!      FAMILY:         TTL
!
!      POWER:          VCC = P-16, GND = P-8
!
!      DESCRIPTION:     DUAL 4-LINE-TO-1-LINE DATA
!                      SELECTORS/MULTIPLEXERS.
!
!      UNUSED PINS:     NONE
!

```

```

!      FUNCTIONS:
!

```

```

      INV(SBAR,.G1);
      INV(IAB,.A+2);
      INV(IA,IAB+4);

```

```

!      G-SBAR1 = INV(P-1)
!      G-IAB = NOT(P-14)
!      G-IA = INV(G-IAB)

```

```

      INV(IBB,.B);
      INV(IB,IBB);

```

```

!      G-IBB = NOT(P-2)
!      G-IB = INV(G-IBB)
!

```

```

!      AND4(O1,SBAR,IAB,IBB,.I10);
!      G-O1A = AND(G-SBAR1, P-6, G-IAB, G-IBB)

```

```

      AND4(O2,SBAR,IA,IBB,.I11);
!   G-O2A = AND(G-SBAR1, P-5, G-IA, G-IBB)

      AND4(O3,SBAR,IAB,IB,.I12);
!   G-O3A = AND(G-SBAR1, P-4, G-IAB, G-IB)

      AND4(O4,SBAR,IA,IB,.I13);
!   G-O4A = AND(G-SBAR1, P-3, G-IA, G-IB)

      ORR4(OUT,O1,O2,O3,O4);
      .R1 = NOT .OUT;

!   G-OUT1(P-7) = NOR(G-O1A, G-O2A, G-O3A, G-O4A)
!
```

```

      INV(SBAR,.G2);
!   G-SBAR2 = INV(P-15)
!

      AND4(O1,SBAR,IAB,IBB,.I20);
!   G-O1B = AND(G-SBAR2, P-10, G-IAB, G-IBB)

      AND4(O2,SBAR,IA,IBB,.I21);
!   G-O2B = AND(G-SBAR2, P-11, G-IA, G-IBB)

      AND4(O3,SBAR,IAB,IB,.I22);
!   G-O3B = AND(G-SBAR2, P-12, G-IAB, G-IB)

      AND4(O4,SBAR,IA,IB,.I23);
!   G-O4B = AND(G-SBAR2, P-13, G-IA, G-IB)

      ORR4(OUT,O1,O2,O3,O4);
      .R2 = NOT .OUT;

!   G-OUT2(P-9) = NOR(G-O1B, G-O2B, G-O3B, G-O4B)
!

!   $ END CHIP $

END % ( routine tc352 )% ;
```

```

!   NOTE: SOURCE DATA - "THE TTL DATA BOOK
!   FOR DESIGN ENGINEERS",
!   SECOND EDITION

END

ELUDDOM
```

TABLE 42 (CONT'D)

```

MODULE TC374(language(%bliss36(bliss36) %bliss32(bliss32)),
  addressing_mode(external = long_relative,
    nonexternal = long_relative) ) =
BEGIN  %( module tcs )%

require 'RTNEST,R32';

```

```

GLOBAL ROUTINE TC374(OC,Q1,D1,D2,Q2,Q3,D3,D4,Q4,K,
  CLOCK,Q5,D5,D6,Q6,Q7,D7,D8,Q8,J) : novalue =
BEGIN  %( routine tc374 )%

```

```

  OWN CLB,OCB,A;

```

```

!           $ CHIP DEFINITION $
!
!
! TYPE:           54_LS_374
!
! FAMILY:         TTL
!
! POWER:          VCC = P-20, GND = P-10
!
! DESCRIPTION:    OCTAL D-TYPE TRANSPARENT,
!                 LATCHES, AND EDGE-TRIGGERED
!                 FLIP-FLOPS WITH 3-STATE OUTPUTS.
!
! UNUSED PINS:    NONE
!
!
! FUNCTIONS:
!
!   BUF(CLB,.CLOCK); BUF(OCB,.OC);
!
!   G-OCB = INV(P-1
!   G-CLB = NOT(P-11)
!
!
!   ST[.K] = (..D1 AND .CLB) OR (.ST[.K] AND NOT .CLB);
!
!
!   FF-Q1 = D(DATA=P-3,
!           CLK_UP=G-CLB)
!
!   ORR(.Q1,OCB,ST[.K]);
!   G-Q1(P-2) = AND(INV(FF-Q1')); DIS_LOW(G-OCB)
!
!   ST[.K+2] = (..D2 AND .CLB) OR (.ST[.K+2] AND NOT .CLB);

```


END %(routine tc374)%;

! \$ END CHIP \$

!
!
! NOTE: SOURCE DATA - "THE TTL DATA BOOK
FOR DESIGN ENGINEERS",
SECOND EDITION

END

ELUDOM

TABLE 43 (CONT'D)

```

!      FF-Q2 = D(DATA=P-4,
!                CLK_UP=G-CLB)

      ORR(.Q2, OCB, ST[.K+2]);
!      G-Q2(P-5) = AND(INV(FF-Q2'));DIS_LOW(G-OCB)
!

      ST[.K+4] = (.D3 AND .CLB) OR (.ST[.K+4] AND NOT .CLB);

!
!      FF-Q3 = D(DATA=P-7,
!                CLK_UP=G-CLB)

      ORR(.Q3, OCB, ST[.K+4]);
!      G-Q3(P-6) = AND(INV(FF-Q3'));DIS_LOW(G-OCB)
!

      ST[.K+6] = (.D4 AND .CLB) OR (.ST[.K+6] AND NOT .CLB);

!
!      FF-Q4 = D(DATA=P-8,
!                CLK_UP=G-CLB)

      ORR(.Q4, OCB, ST[.K+6]);
!      G-Q4(P-9) = AND(INV(FF-Q4'));DIS_LOW(G-OCB)
!

      ST[.K+8] = (.D5 AND .CLB) OR (.ST[.K+8] AND NOT .CLB);

!
!      FF-Q5 = D(DATA=P-13,
!                CLK_UP=G-CLB)

      ORR(.Q5, OCB, ST[.K+8]);
!      G-Q5(P-12) = AND(INV(FF-Q5'));DIS_LOW(G-OCB)
!

      ST[.K+10] = (.D6 AND .CLB) OR (.ST[.K+10] AND NOT .CLB);

!
!      FF-Q6 = D(DATA=P-14,
!                CLK_UP=G-CLB)

      ORR(.Q6, OCB, ST[.K+10]);
!      G-Q6(P-15) = AND(INV(FF-Q6'));DIS_LOW(G-OCB)
!

      ST[.K+12] = (.D7 AND .CLB) OR (.ST[.K+12] AND NOT .CLB);

!
!      FF-Q7 = D(DATA=P-17,
!                CLK_UP=G-CLB)

      ORR(.Q7, OCB, ST[.K+12]);
!      G-Q7(P-16) = AND(INV(FF-Q7'));DIS_LOW(G-OCB)
!

      ST[.K+14] = (.D8 AND .CLB) OR (.ST[.K+14] AND NOT .CLB);

!
!      FF-Q8 = D(DATA=P-18,
!                CLK_UP=G-CLB)

      ORR(.Q8, OCB, ST[.K+14]);
!      G-Q8(P-19) = AND(INV(FF-Q8'));DIS_LO(G-OCB)
!
!

```

```

MODULE TC377(language(%bliss36(bliss36) %bliss32(bliss32)),
addressing_mode(external = long_relative,
nonexternal = long_relative) ) =
BEGIN  %( module tcs )%

```

```

require 'RTNEST,R32';

```

```

MACRO ORR4(R,I1,I2,I3,I4,J) =
      R=(.I1 OR .I2 OR .I3 OR .I4) FAND(J,0)
      FORK(J,1)%;

```

```

GLOBAL ROUTINE TC377(GBAR,Q1,D1,D2,Q2,Q3,D3,D4,Q4,K,
CLOCK,Q5,D5,D6,Q6,Q7,D7,D8,Q8,J) : novalue =
BEGIN %( routine tc377 )%

```

```

LOCAL CLB,A,B,C;

```

```

!           $ CHIP DEFINITION $
!
!
! TYPE:           25_LS_377
!
! FAMILY:         TTL
!
! POWER:          VCC = P-20, GND = P-10
!
! DESCRIPTION:    OCTAL D-TYPE FLIP-FLOPS.
!
! UNUSED PINS:    NONE
!
!
! FUNCTIONS:
!
!     BUF(CLB,.CLOCK);
!     B = NOT .GBAR;
!     ANDR(C,B,CLB);
!
! G-CLB = NOT(P-11)
! G-B = INV(P-1)
!
! FF=ENBL = D(DATA=INV(G-B),
!             CLK_UP=G-CLB,
!             CLEAR=INV(G-CLB))
!
!     ANDR(B,C,CLB);
!     ST[.K] = (.D1 AND .B) OR (.ST[.K] AND NOT .B);
!     .Q1 = .ST[.K];

```

```

!
!
!
FF-Q1(P-2) = D(DATA=P-3,
                CLK_UP=FF-ENBL)

    ANDR(B,C,CLB);
    ST[.K+2] = (..D2 AND .B) OR (.ST[.K+2] AND NOT .B);
    .Q2 = .ST[.K+2];

!
!
!
FF-Q2(P-5) = D(DATA=P-4,
                CLK_UP=FF-ENBL)

    ANDR(B,C,CLB);
    ST[.K+4] = (..D3 AND .B) OR (.ST[.K+4] AND NOT .B);
    .Q3 = .ST[.K+4];

!
!
!
FF-Q3(P-6) = D(DATA=P-7,
                CLK_UP=FF-ENBL)

    ANDR(B,C,CLB);
    ST[.K+6] = (..D4 AND .B) OR (.ST[.K+6] AND NOT .B);
    .Q4 = .ST[.K+6];

!
!
!
FF-Q4(P-9) = D(DATA=P-8,
                CLK_UP=FF-ENBL)

    ANDR(B,C,CLB);
    ST[.K+8] = (..D5 AND .B) OR (.ST[.K+8] AND NOT .B);
    .Q5 = .ST[.K+8];

!
!
!
FF-Q5(P-12) = D(DATA=P-13,
                 CLK_UP=FF-ENBL)

    ANDR(B,C,CLB);
    ST[.K+10] = (..D6 AND .B) OR (.ST[.K+10] AND NOT .B);
    .Q6 = .ST[.K+10];

!
!
!
FF-Q6(P-15) = D(DATA=P-14,
                 CLK_UP=FF-ENBL)

    ANDR(B,C,CLB);
    ST[.K+12] = (..D7 AND .B) OR (.ST[.K+12] AND NOT .B);
    .Q7 = .ST[.K+12];

!
!
!
FF-Q7(P-16) = D(DATA=P-17,
                 CLK_UP=FF-ENBL)

    ANDR(B,C,CLB);
    ST[.K+14] = (..D8 AND .B) OR (.ST[.K+14] AND NOT .B);
    .Q8 = .ST[.K+14];

!
!
!
FF-Q8(P-19) = D(DATA=P-18,
                 CLK_UP=FF-ENBL)

END %( routine tc377 )%;

!
$ END CHIP $

```

! ! ! ! !
END ELUDOM

NOTE: SOURCE DATA - "THE TTL DATA BOOK
FOR DESIGN ENGINEERS",
SECOND EDITION
AND MODIFIED DWG. ENCLOSED

```

MODULE T9407(language( %bliss36(bliss36) %bliss32(bliss32)),
    addressing_mode(external = long_relative,
        nonexternal = long_relative) ) =
BEGIN    %( module t9407 )%

```

```

    REQUIRE "RTNEST,R32";
    GLOBAL ROUTINE TC9407(EXN,I0,I1,I2,I3,EOXN,CLK,X0,X1,X2,X3,K,
        CON,O3N,D3N,O2N,D2N,O1N,D1N,O0N,D0N,EOO,CIN,J) :
        novalue =

```

```

BEGIN    %( routine tc9407 )%
    OWN    CLOCK,TSEL,PSEL,I2B,S0,S1,S2,S3,Y0,Y1,Y2,Y3;

    OWN    A,B,C,D,E,F,G,T,P0,P1,P2,P3,G0,G1,G2,G3,CI,X;
    OWN    CLR,CLR;

```

```

    MACRO A0=ST[.K]%,A1=ST[.K+1]%,A2=ST[.K+2]%,A3=ST[.K+3]%,
        B0=ST[.K+4]%,B1=ST[.K+5]%,B2=ST[.K+6]%,B3=ST[.K+7]%,
        K1=,K+8%,K2=,K+16%,K3=,K+24%;

```

```

    T = -1;

```

```

!           $ CHIP DEFINITION $
!
!           TYPE:           9407
!
!           FAMILY:         TTL
!
!           POWER:          VCC = P-24, GND = P-12
!
!           DESCRIPTION:    MEMORY ADDRESS PROCESSOR.
!
!           UNUSED PINS:    NONE
!
!           FUNCTIONS:
!
!           INV(ST[.K],.I2);
!           INV(A,ST[.K]);
!           INV(B,A);
!           BUF(OCN,.EOXN);
!           G-OCN = INV(P-6)
!           G-I2B = NOT(P-4)
!           G-SELA = NOT(G-I2B)
!           G-SELB = INV(G-SELA)
!
!           ANDR(C,A0,A);
!           ANDR(D,B0,B);
!           G-G1A = AND(FF-A0, G-SELA)
!           G-G1B = AND(FF-B0, G-SELB)

```

```

!      ORR(E,C,D);
!      ORR(.Y1,E,OCN);
!      GTS-Y1(P=8) = OR(G-G1A, G-G1B);DIS_LOW(G-OCN)

ANDR(C,A1,A);
ANDR(D,B1,B);
!      G-G2A = AND(FF-A1, G-SELA)
!      G-G2B = AND(FF-B1, G-SELB)

ORR(E,C,D);
ORR(.Y2,E,OCN);
!      GTS-Y2(P=9) = OR(G-G2A, G-G2B);DIS_LOW(G-OCN)

ANDR(C,A2,A);
ANDR(D,B2,B);
!      G-G3A = AND(FF-A2, G-SELA)
!      G-G3B = AND(FF-B2, G-SELB)

ORR(E,C,D);
ORR(.Y3,E,OCN);
!      GTS-Y3(P=10) = OR(G-G3A, G-G3B);DIS_LOW(G-OCN)

ANDR(C,A3,A);
ANDR(D,B3,B);
!      G-G4A = AND(FF-A3, G-SELA)
!      G-G4B = AND(FF-B3, G-SELB)

ORR(E,C,D);
ORR(.Y4,E,OCN);
!      GTS-Y4(P=11) = OR(G-G4A, G-G4B);DIS_LOW(G-OCN)

I2B = .ST(.K+32);
NAND(A,.I3,I2B);

!      G-A = NAND(P=5, G-I2B)

NAND(B,A,.I1);
!      G-B = NAND(G-A, P=3)

NAND(C,B,T);
!      G-C = NOT(G-B)

F = NOT .CLK;      NOR(CLOCK, F, .EXN);
!      G-CLOCK = OR(P=7, P=1)

F = NOT .CLOCK;      nor(TSEL, f, C);
!      G-TSEL = OR(G-CLOCK, G-C)

F = NOT .CLOCK;      NOR(PSEL, F, B);
!      G-PSEL = OR(G-CLOCK, G-B)

INV(A,I2B);
!      G-I2B' = NOT(G-I2B)

NOR(B,A,.I3);

```

```

!           G-SY = AND(INV(G-I2B'), INV(P-5))
NOR(A,I2B,.I3);
!           G-SX = AND(INV(G-I2B), INV(P-5))

ANDR(C,A0,A);
!           G-A0 = AND(FF-A0, G-SX)

ANDR(D,B0,B);
!           G-B0 = AND(FF-B0, G-SY)

NOR(Y0,C,D);
!           G-Y0 = NOR(G-A1, G-B1)

ANDR(C,A1,A);
!           G-A1 = AND(FF-A1, G-SX)

ANDR(D,B1,B);
!           G-B1 = AND(FF-B1, G-SY)

NOR(Y1,C,D);
!           G-Y1 = NOR(G-A2, G-B2)

ANDR(C,A2,A);
!           G-A2 = AND(FF-A2, G-SX)

ANDR(D,B2,B);
!           G-B2 = AND(FF-B2, G-SY)

NOR(Y2,C,D);
!           G-Y4 = NOR(G-A3, G-B3)

ANDR(C,A3,A);
!           G-A3 = AND(FF-A3, G-SX)

ANDR(D,B3,B);
!           G-B3 = AND(FF-B3, G-SY)

NOR(Y3,C,D);
!           G-Y3 = NOR(G-A4, G-B4)
!
!
!
!
!           NAND(P0, .DON, Y0);   NOR(G0, .DON, Y0);
!           G-G0 = NOR(P-21, G-Y0)
!           G-P0 = NAND(P-21, G-Y0)

!           NAND(P1, .D1N, Y1);   NOR(G1, .D1N, Y1);
!           G-G1 = NOR(P-19, G-Y1)
!           G-P1 = NAND(P-19, G-Y1)

!           NAND(P2, .D2N, Y2);   NOR(G2, .D2N, Y2);
!           G-G2 = NOR(P-17, G-Y2)
!           G-P2 = NAND(P-17, G-Y2)

!           NAND(P3, .D3N, Y3);   NOR(G3, .D3N, Y3);
!           G-G3 = NOR(P-15, G-Y3)
!           G-P3 = NAND(P-15, G-Y3)
!
!
!

```



```

INV(CI,.CIN); INV(A,CI);
G-CI = NOT(P-23)
G-C1 = NOT(G-CI)

F = NOT ,G0; ANDR(G, F, P0); XORR(S0, G, A);
G-C2 = AND(INV(G-G0), G-P0)
G-S0 = XOR(G-C1, G-C2)

BUF(A,G0); ANDR(B,P0,CI); NOR(C,A,B);
G-D2 = AND(G-G0)
G-D1 = AND(G-P0, G-CI)
G-D4 = NOR(G-D1, G-D2)

F = NOT ,G1; ANDR(G, F, P1); XORR(S1, G, C);
G-D3 = AND(INV(G-G1), G-P1)
G-S1 = XOR(G-D3, G-D4)

BUF(A,G1); ANDR(B,G0,P1);
C = AND3(P1,P0,CI); NOR3(D,A,B,C);
G-E3 = AND(G-G1)
G-E2 = AND(G-G0, G-P1)
G-E1 = AND(G-P1, G-P0, G-CI)
G-E5 = NOR(G-E1, G-E2, G-E3)

F = NOT ,G2; ANDR(G, F, P2); XORR(S2, G, D);
G-E4 = AND(INV(G-G2), G-P2)
G-S2 = XOR(G-E4, G-E5)

BUF(A,G2); ANDR(B,G1,P2);
C = AND3(P2,P1,G0);
AND4(D,P2,P1,P0,CI); NOR4(E,A,B,C,D);
G-F4 = AND(G-G2)
G-F3 = AND(G-G1, G-P2)
G-F2 = AND(G-P2, G-P1, G-G0)
G-F1 = AND(P-P2, P-P1, P-P0, P-CI)
G-F6 = NOR(G-F1, G-F2, G-F3, G-F4)

F = NOT ,G3; ANDR(G, F, P3); XORR(S3, G, E);
G-F5 = AND(INV(G-G3), G-P3)
G-S3 = XOR(G-F5, G-F6)

INV(A,G3); NAND(B,G2,P3);
G-H5 = INV(G-G3)
G-H4 = NAND(G-G2, G-P3)

NAND3(C,P3,P2,G1); NAND4(D,P3,P2,P1,G0);
G-H3 = NAND(G-P3, G-P2, G-G1)
G-H2 = NAND(G-P3, G-P2, G-P1, G-G0)

NAND5(E,P3,P2,P1,P0,CI); AND5(.CON,A,B,C,D,E);
G-H1 = NAND(G-P3, G-P2, G-P1, G-P0, G-CI)

G-CO'(P-13) = AND(G-H1, G-H2, G-H3, G-H4, G-H5)

BUF(PCLB,PSEL);
G-CLB = AND(G-PSEL)

```

```

!           FF-A0      =      D(DATA = G-S0,
!                               CLK_DOWN = G-PCLB)

A1 = (NOT .S1 AND .CLB) OR (.A1 AND NOT .CLB);

!           FF-A1      =      D(DATA = G-S1,
!                               CLK_DOWN = G-PCLB)

A2 = (NOT .S2 AND .CLB) OR (.A2 AND NOT .CLB);

!           FF-A2      =      D(DATA = G-S2,
!                               CLK_DOWN = G-PCLB)

A3 = (NOT .S3 AND .CLB) OR (.A3 AND NOT .CLB);

!           FF-A3      =      D(DATA = G-S3,
!                               CLK_DOWN = G-PCLB)

!
!           BUF(CLB,TSEL);
!           G-TCLB = AND(G-TSEL)

B0 = (NOT .S0 AND .CLB) OR (.B0 AND NOT .CLB);

!           FF-B0      =      D(DATA = G-S0,
!                               CLK_DOWN = G-TCLB)

B1 = (NOT .S1 AND .CLB) OR (.B1 AND NOT .CLB);

!           FF-B1      =      D(DATA = G-S1,
!                               CLK_DOWN = G-TCLB)

B2 = (NOT .S2 AND .CLB) OR (.B2 AND NOT .CLB);

!           FF-B2      =      D(DATA = G-S2,
!                               CLK_DOWN = G-TCLB)

B3 = (NOT .S3 AND .CLB) OR (.B3 AND NOT .CLB);

!           FF-B3      =      D(DATA = G-S3,
!                               CLK_DOWN = G-TCLB)

!
!           BUF(CLB,CLOCK);
!           G-OCLB = AND(CLOCK)

ST[K3] = (.S0 AND .CLB) OR (.ST[K3] AND NOT .CLB);
.O0N = .ST[K3];

!           FF-O0(;P-20) = D(DATA = G-S0,
!                               CLK_DOWN = G-OCLB)

ST[K3+2] = (.S1 AND .CLB) OR (.ST[K3+2] AND NOT .CLB);
.O1N = .ST[K3+2];

!           FF-O1(;P-18) = D(DATA = G-S1,
!                               CLK_DOWN = G-OCLB)

```

```
ST[K3+4] = (.S2 AND .CLB) OR (.ST[K3+4] AND NOT .CLB);
.O2N = .ST[K3+4];
```

```
!
!
```

```
FF-02(;P=16) = D(DATA = G-S2,
                  CLK_DOWN = G-OCLB)
```

```
ST[K3+6] = (.S3 AND .CLB) OR (.ST[K3+6] AND NOT .CLB);
.O3N = .ST[K3+6];
```

```
!
!
!
!
!
```

```
FF-03(;P=14) = D(DATA = G-S3,
                  CLK_DOWN = G-OCLB)
```

```
$ END CHIP $
```

```
END      %( ROUTINE TC9407 )%;
END      %( MODULE  T9407  )%;
ELUDOM
```

TABLE 45 (CONT'D)

```

MODULE T2901A(language (%bliss36(bliss36)
                    %bliss32(bliss32)),
              addressing_mode(external = long_relative,
                              nonexternal = long_relative) ) =

```

```

BEGIN

```

```

    require 'RTNEST.R32';

```

```

    GLOBAL ROUTINE TC2901A(A3,A2,A1,A0,I6,I8,I7,RAM3,
        RAM0,J,FEO,I0,I1,I2,CP,Q3,B0,B1,B2,B3,Q0,
        D3,D2,D1,D0,I3,I5,I4,CN,K,F3P,GBAR,CN4,
        OVR,PBAR,Y0,Y1,Y2,Y3,CBAR) : novalue =
    BEGIN
        REQUIRE 'BINDM.R32';

```

```

! FOUR BIT SLICE MICROPROCESSOR. THE INPUTS ARE IN PIN ORDER
! WITH J AND K FOR VCC AND GND. THE VARIABLE NAMES
! FOR THE LOCAL VARIABLES CORRESPOND TO THE MARKINGS ON
! THE AND SCHEMATIC DIAGRAM WITH A SUFFIX OF B INDICATING
! AN INVERSION.

```

```

own IOB,I1B,I2B,I2BB,ID,ROB,SOB,R1B,S1B,R2B,S2B,R3B,
    S3B,CR,CS,FC,INH,P0,G0,P1,G1,P2,G2,P3,G3,CAR0,CAR1,
    CAR2,CAR3,F0,F1,F2,F3,TEMP,A,B,
    RE,SF,I6B,I7B,I6BB,CLBB,SFB,CE;

```

```

own AO : vector[4],
    BO : vector[4];

```

```

LOCAL

```

```

    AMULT: vector[16];

```

```

LOCAL

```

```

    IA, IAB, IB, IBB, IC, ICB, IDD, IDB;

```

```

    $ CHIP DEFINITION $

```

```

TYPE:          AM_2901_A

```

```

FAMILY:        TTL

```

```

POWER:         VCC = P-10, GND = P-30

```

```

DESCRIPTION:    BIPOLAR MICROCONTROLLER.

```

```

UNUSED PINS:    NONE

```

```

FUNCTIONS:

```

```

    ! CLOCK LOGIC

```

```

      INV(CLB,,CP); INV(CLBB,CLB);
      !
      !      G-CLB = NOT(P-15)
      !      G-CLBB = NOT(G-CLB)
      !
      ! READ RAM OUTPUTS AND SET INPUT FOR LATCHES
      !
      !
      !
      !      INV(IAB,,A0); INV(IA,IAB);
      !      !      G-IAB1 = NOT(P-4)
      !      !      G-IA1 = INV(G-IAB1)
      !
      !      INV(IBB,,A1); INV(IB,IBB);
      !      !      G-IBB1 = NOT(P-3)
      !      !      G-IB1 = INV(G-IBB1)
      !
      !      INV(ICB,,A2); INV(IC,ICB);
      !      !      G-ICB1 = NOT(P-2)
      !      !      G-IC1 = INV(G-ICB1)
      !
      !      INV(IDB,,A3); INV(IDD,IDB);
      !      !      G-IDB1 = NOT(P-1)
      !      !      G-IDD1 = INV(G-IDB1)
      !
      !      AND4(AMULT,IAB,IBB,ICB,IDB);
      !      !      G-AMULT0 = AND(G-IAB1, G-IBB1, G-ICB1, G-IDB1)
      !
      !      AND4((AMULT+1*%upval),IA,IBB,ICB,IDB);
      !      !      G-AMULT1 = AND(G-IA1, G-IBB1, G-ICB1, G-IDB1)
      !
      !      AND4((AMULT+2*%upval),IAB,IB,ICB,IDB);
      !      !      G-AMULT2 = AND(G-IAB1, G-IB1, G-ICB1, G-IDB1)
      !
      !      AND4((AMULT+3*%upval),IA,IB,ICB,IDB);
      !      !      G-AMULT3 = AND(G-IA1, G-IB1, G-ICB1, G-IDB1)
      !
      !      AND4((AMULT+4*%upval),IAB,IBB,IC,IDB);
      !      !      G-AMULT4 = AND(G-IAB1, G-IBB1, G-IC1, G-IDB1)
      !
      !      AND4((AMULT+5*%upval),IA,IBB,IC,IDB);
      !      !      G-AMULT5 = AND(G-IA1, G-IBB1, G-IC1, G-IDB1)
      !
      !      AND4((AMULT+6*%upval),IAB,IB,IC,IDB);
      !      !      G-AMULT6 = AND(G-IAB1, G-IB1, G-IC1, G-IDB1)
      !
      !      AND4((AMULT+7*%upval),IA,IB,IC,IDB);
      !      !      G-AMULT7 = AND(G-IA1, G-IB1, G-IC1, G-IDB1)
      !
      !      AND4((AMULT+8*%upval),IAB,IBB,ICB,IDD);
      !      !      G-AMULT8 = AND(G-IAB1, G-IBB1, G-ICB1, G-IDD1)
      !
      !      AND4((AMULT+9*%upval),IA,IBB,ICB,IDD);
      !      !      G-AMULT9 = AND(G-IA1, G-IBB1, G-ICB1, G-IDD1)
      !
      !      AND4((AMULT+10*%upval),IAB,IB,ICB,IDD);
      !      !      G-AMULT10 = AND(G-IAB1, G-IB1, G-ICB1, G-IDD1)
      !
      !      AND4((AMULT+11*%upval),IA,IB,ICB,IDD);
      !      !      G-AMULT11 = AND(G-IA1, G-IB1, G-ICB1, G-IDD1)

```

```

AND4((AMULT+12*%upval),IAB,IBB,IC,IDD);
!      G-AMULT12 = AND(G-IAB1, G-IBB1, G-IC1, G-IDD1)

AND4((AMULT+13*%upval),IA,IBB,IC,IDD);
!      G-AMULT13 = AND(G-IA1, G-IBB1, G-IC1, G-IDD1)

AND4((AMULT+14*%upval),IAB,IB,IC,IDD);
!      G-AMULT14 = AND(G-IAB1, G-IB1, G-IC1, G-IDD1)

AND4((AMULT+15*%upval),IA,IB,IC,IDD);
!      G-AMULT15 = AND(G-IA1, G-IB1, G-IC1, G-IDD1)
!
!
!
INV(IAB,.B0); INV(IA,IAB);
!      G-IAB2 = NOT(P-17)
!      G-IA2 = INV(G-IAB2)

INV(IBB,.B1); INV(IB,IBB);
!      G-IBB2 = NOT(P-18)
!      G-IB2 = INV(G-IBB2)

INV(ICB,.B2); INV(IC,ICB);
!      G-ICB2 = NOT(P-19)
!      G-IC2 = INV(G-ICB2)

INV(IDB,.B3); INV(IDD,IDB);
!      G-IDB2 = NOT(P-20)
!      G-IDD2 = INV(G-IDB2)

AND4(BMULT,IAB,IBB,ICB,IDB);
!      G-BMULT0 = AND(G-IAB2, G-IBB2, G-ICB2, G-IDB2)

AND4((BMULT+1*%upval),IA,IBB,ICB,IDB);
!      G-BMULT1 = AND(G-IA2, G-IBB2, G-ICB2, G-IDB2)

AND4((BMULT+2*%upval),IAB,IB,ICB,IDB);
!      G-BMULT2 = AND(G-IAB2, G-IB2, G-ICB2, G-IDB2)

AND4((BMULT+3*%upval),IA,IB,ICB,IDB);
!      G-BMULT3 = AND(G-IA2, G-IB2, G-ICB2, G-IDB2)

AND4((BMULT+4*%upval),IAB,IBB,IC,IDB);
!      G-BMULT4 = AND(G-IAB2, G-IBB2, G-IC2, G-IDB2)

AND4((BMULT+5*%upval),IA,IBB,IC,IDB);
!      G-BMULT5 = AND(G-IA2, G-IBB2, G-IC2, G-IDB2)

AND4((BMULT+6*%upval),IAB,IB,IC,IDB);
!      G-BMULT6 = AND(G-IAB2, G-IB2, G-IC2, G-IDB2)

AND4((BMULT+7*%upval),IA,IB,IC,IDB);
!      G-BMULT7 = AND(G-IA2, G-IB2, G-IC2, G-IDB2)

AND4((BMULT+8*%upval),IAB,IBB,ICB,IDD);
!      G-BMULT8 = AND(G-IAB2, G-IBB2, G-ICB2, G-IDD2)

AND4((BMULT+9*%upval),IA,IBB,ICB,IDD);
!      G-BMULT9 = AND(G-IA2, G-IBB2, G-ICB2, G-IDD2)

AND4((BMULT+10*%upval),IAB,IB,ICB,IDD);

```

```

!           G-BMULT10   = AND(G-IAB2, G-IB2, G-ICB2, G-IDD2)
      AND4((BMULT+11*%upval),IA,IB,ICB,IDD);
!           G-BMULT11   = AND(G-IA2, G-IB2, G-ICB2, G-IDD2)
      AND4((BMULT+12*%upval),IAB,IBB,IC,IDD);
!           G-BMULT12   = AND(G-IAB2, G-IBB2, G-IC2, G-IDD2)
      AND4((BMULT+13*%upval),IA,IBB,IC,IDD);
!           G-BMULT13   = AND(G-IA2, G-IBB2, G-IC2, G-IDD2)
      AND4((BMULT+14*%upval),IAB,IB,IC,IDD);
!           G-BMULT14   = AND(G-IAB2, G-IB2, G-IC2, G-IDD2)
      AND4((BMULT+15*%upval),IA,IB,IC,IDD);
!           G-BMULT15   = AND(G-IA2, G-IB2, G-IC2, G-IDD2)
!
!
!

```

(

```

AO = ANDF((AMULT), (RMO));
!       G-A00 = AND(FF-RM00); DIS_HIGH(G-AMULT0)

AO = ,AO OR ANDF((AMULT+4), (RMO+4));

!       G-A00 = AND(FF-RM01); DIS_HIGH(G-AMULT1)
AO = ,AO OR ANDF((AMULT+8), (RMO+8));
!       G-A00 = AND(FF-RM02); DIS_HIGH(G-AMULT2)

AO = ,AO OR ANDF((AMULT+12), (RMO+12));
!       G-A00 = AND(FF-RM03); DIS_HIGH(G-AMULT3)

AO = ,AO OR ANDF((AMULT+16), (RMO+16));
!       G-A00 = AND(FF-RM04); DIS_HIGH(G-AMULT4)

AO = ,AO OR ANDF((AMULT+20), (RMO+20));
!       G-A00 = AND(FF-RM05); DIS_HIGH(G-AMULT5)

AO = ,AO OR ANDF((AMULT+24), (RMO+24));
!       G-A00 = AND(FF-RM06); DIS_HIGH(G-AMULT6)

AO = ,AO OR ANDF((AMULT+28), (RMO+28));
!       G-A00 = AND(FF-RM07); DIS_HIGH(G-AMULT7)

```

```

AO = ,AO OR ANDF((AMULT+32), (RMO+32));
!      G-A00 = AND(FF-RM08); DIS_HIGH(G-AMULT8)

AO = ,AO OR ANDF((AMULT+36), (RMO+36));
!      G-A00 = AND(FF-RM09); DIS_HIGH(G-AMULT9)

AO = ,AO OR ANDF((AMULT+40), (RMO+40));
!      G-A00 = AND(FF-RM010); DIS_HIGH(G-AMULT10)

AO = ,AO OR ANDF((AMULT+44), (RMO+44));
!      G-A00 = AND(FF-RM011); DIS_HIGH(G-AMULT11)

AO = ,AO OR ANDF((AMULT+48), (RMO+48));
!      G-A00 = AND(FF-RM012); DIS_HIGH(G-AMULT12)

AO = ,AO OR ANDF((AMULT+52), (RMO+52));
!      G-A00 = AND(FF-RM013); DIS_HIGH(G-AMULT13)

AO = ,AO OR ANDF((AMULT+56), (RMO+56));
!      G-A00 = AND(FF-RM014); DIS_HIGH(G-AMULT14)

AO = ,AO OR ANDF((AMULT+60), (RMO+60));
!      G-A00 = AND(FF-RM015); DIS_HIGH(G-AMULT15)

AO+4 = ANDF((AMULT), (RMO+64));
!      G-A01 = AND(FF-RM016); DIS_HIGH(G-AMULT0)

AO+4 = ,(AO+4) OR ANDF((AMULT+4), (RMO+68));
!      G-A01 = AND(FF-RM017); DIS_HIGH(G-AMULT1)

AO+4 = ,(AO+4) OR ANDF((AMULT+8), (RMO+72));
!      G-A01 = AND(FF-RM018); DIS_HIGH(G-AMULT2)

AO+4 = ,(AO+4) OR ANDF((AMULT+12), (RMO+76));
!      G-A01 = AND(FF-RM019); DIS_HIGH(G-AMULT3)

```



```

AO+4 = ,(AO+4) OR ANDF((AMULT+16), (RMO+80));
!      G-AO1 = AND(FF-RMO20); DIS_HIGH(G-AMULT4)

AO+4 = ,(AO+4) OR ANDF((AMULT+20), (RMO+84));
!      G-AO1 = AND(FF-RMO21); DIS_HIGH(G-AMULT5)

AO+4 = ,(AO+4) OR ANDF((AMULT+24), (RMO+88));
!      G-AO1 = AND(FF-RMO22); DIS_HIGH(G-AMULT6)

AO+4 = ,(AO+4) OR ANDF((AMULT+28), (RMO+92));
!      G-AO1 = AND(FF-RMO23); DIS_HIGH(G-AMULT7)

AO+4 = ,(AO+4) OR ANDF((AMULT+32), (RMO+96));
!      G-AO1 = AND(FF-RMO24); DIS_HIGH(G-AMULT8)

AO+4 = ,(AO+4) OR ANDF((AMULT+36), (RMO+100));
!      G-AO1 = AND(FF-RMO25); DIS_HIGH(G-AMULT9)

AO+4 = ,(AO+4) OR ANDF((AMULT+40), (RMO+104));
!      G-AO1 = AND(FF-RMO26); DIS_HIGH(G-AMULT10)

AO+4 = ,(AO+4) OR ANDF((AMULT+44), (RMO+108));
!      G-AO1 = AND(FF-RMO27); DIS_HIGH(G-AMULT11)

AO+4 = ,(AO+4) OR ANDF((AMULT+48), (RMO+112));
!      G-AO1 = AND(FF-RMO28); DIS_HIGH(G-AMULT12)

AO+4 = ,(AO+4) OR ANDF((AMULT+52), (RMO+116));
!      G-AO1 = AND(FF-RMO29); DIS_HIGH(G-AMULT13)

AO+4 = ,(AO+4) OR ANDF((AMULT+56), (RMO+120));
!      G-AO1 = AND(FF-RMO30); DIS_HIGH(G-AMULT14)

AO+4 = ,(AO+4) OR ANDF((AMULT+60), (RMO+124));
!      G-AO1 = AND(FF-RMO31); DIS_HIGH(G-AMULT15)

```

```

AO+8 = ANDF((AMULT), (RMO+128));
!      G-AO2 = AND(FF-RMO32); DIS_HIGH(G-AMULT0)
AO+8 = ,(AO+8) OR ANDF((AMULT+4), (RMO+132));
!      G-AO2 = AND(FF-RMO33); DIS_HIGH(G-AMULT1)
AO+8 = ,(AO+8) OR ANDF((AMULT+8), (RMO+136));
!      G-AO2 = AND(FF-RMO34); DIS_HIGH(G-AMULT2)
AO+8 = ,(AO+8) OR ANDF((AMULT+12), (RMO+140));
!      G-AO2 = AND(FF-RMO35); DIS_HIGH(G-AMULT3)
AO+8 = ,(AO+8) OR ANDF((AMULT+16), (RMO+144));
!      G-AO2 = AND(FF-RMO36); DIS_HIGH(G-AMULT4)
AO+8 = ,(AO+8) OR ANDF((AMULT+20), (RMO+148));
!      G-AO2 = AND(FF-RMO37); DIS_HIGH(G-AMULT5)
AO+8 = ,(AO+8) OR ANDF((AMULT+24), (RMO+152));
!      G-AO2 = AND(FF-RMO38); DIS_HIGH(G-AMULT6)
AO+8 = ,(AO+8) OR ANDF((AMULT+28), (RMO+156));
!      G-AO2 = AND(FF-RMO39); DIS_HIGH(G-AMULT7)
AO+8 = ,(AO+8) OR ANDF((AMULT+32), (RMO+160));
!      G-AO2 = AND(FF-RMO40); DIS_HIGH(G-AMULT8)
AO+8 = ,(AO+8) OR ANDF((AMULT+36), (RMO+164));
!      G-AO2 = AND(FF-RMO41); DIS_HIGH(G-AMULT9)
AO+8 = ,(AO+8) OR ANDF((AMULT+40), (RMO+168));
!      G-AO2 = AND(FF-RMO42); DIS_HIGH(G-AMULT10)
AO+8 = ,(AO+8) OR ANDF((AMULT+44), (RMO+172));
!      G-AO2 = AND(FF-RMO43); DIS_HIGH(G-AMULT11)
AO+8 = ,(AO+8) OR ANDF((AMULT+48), (RMO+176));

```

```

!      G-A02 = AND(FF-RM044); DIS_HIGH(G-AMULT12)

AO+8 = ,(AO+8) OR ANDF((AMULT+52), (RM0+180));

!      G-A02 = AND(FF-RM045); DIS_HIGH(G-AMULT13)

AO+8 = ,(AO+8) OR ANDF((AMULT+56), (RM0+184));

!      G-A02 = AND(FF-RM046); DIS_HIGH(G-AMULT14)

AO+8 = ,(AO+8) OR ANDF((AMULT+60), (RM0+188));

!      G-A02 = AND(FF-RM047); DIS_HIGH(G-AMULT15)

AO+12 = ANDF((AMULT), (RM0+192));

!      G-A03 = AND(FF-RM048); DIS_HIGH(G-AMULT0)

AO+12 = ,(AO+12) OR ANDF((AMULT+4), (RM0+196));

!      G-A03 = AND(FF-RM049); DIS_HIGH(G-AMULT1)

AO+12 = ,(AO+12) OR ANDF((AMULT+8), (RM0+200));

!      G-A03 = AND(FF-RM050); DIS_HIGH(G-AMULT2)

AO+12 = ,(AO+12) OR ANDF((AMULT+12), (RM0+204));

!      G-A03 = AND(FF-RM051); DIS_HIGH(G-AMULT3)

AO+12 = ,(AO+12) OR ANDF((AMULT+16), (RM0+208));

!      G-A03 = AND(FF-RM052); DIS_HIGH(G-AMULT4)

AO+12 = ,(AO+12) OR ANDF((AMULT+20), (RM0+212));

!      G-A03 = AND(FF-RM053); DIS_HIGH(G-AMULT5)

AO+12 = ,(AO+12) OR ANDF((AMULT+24), (RM0+216));

!      G-A03 = AND(FF-RM054); DIS_HIGH(G-AMULT6)

AO+12 = ,(AO+12) OR ANDF((AMULT+28), (RM0+220));

!      G-A03 = AND(FF-RM055); DIS_HIGH(G-AMULT7)

AO+12 = ,(AO+12) OR ANDF((AMULT+32), (RM0+224));

!      G-A03 = AND(FF-RM056); DIS_HIGH(G-AMULT8)

```

```

AO+12 = ,(AO+12) OR ANDF((AMULT+36), (RMO+228));
!      G-A03 = AND(FF-RM057); DIS_HIGH(G-AMULT9)

AO+12 = ,(AO+12) OR ANDF((AMULT+40), (RMO+232));
!      G-A03 = AND(FF-RM058); DIS_HIGH(G-AMULT10)

AO+12 = ,(AO+12) OR ANDF((AMULT+44), (RMO+236));
!      G-A03 = AND(FF-RM059); DIS_HIGH(G-AMULT11)

AO+12 = ,(AO+12) OR ANDF((AMULT+48), (RMO+240));
!      G-A03 = AND(FF-RM060); DIS_HIGH(G-AMULT12)

AO+12 = ,(AO+12) OR ANDF((AMULT+52), (RMO+244));
!      G-A03 = AND(FF-RM061); DIS_HIGH(G-AMULT13)

AO+12 = ,(AO+12) OR ANDF((AMULT+56), (RMO+248));
!      G-A03 = AND(FF-RM062); DIS_HIGH(G-AMULT14)

AO+12 = ,(AO+12) OR ANDF((AMULT+60), (RMO+252));
!      G-A03 = AND(FF-RM063); DIS_HIGH(G-AMULT15)


BO = ANDF((BMULT), (RMO));
!      G-B00 = AND(FF-RM00); DIS_HIGH(G-BMULT0)

BO = ,BO OR ANDF((BMULT+4), (RMO+4));
!      G-B00 = AND(FF-RM01); DIS_HIGH(G-BMULT1)

BO = ,BO OR ANDF((BMULT+8), (RMO+8));
!      G-B00 = AND(FF-RM02); DIS_HIGH(G-BMULT2)

BO = ,BO OR ANDF((BMULT+12), (RMO+12));
!      G-B00 = AND(FF-RM03); DIS_HIGH(G-BMULT3)

BO = ,BO OR ANDF((BMULT+16), (RMO+16));
!      G-B00 = AND(FF-RM04); DIS_HIGH(G-BMULT4)

```

```

BO = ,BO OR ANDF((BMULT+20), (RMO+20));
!      G-B00 = AND(FF-RM05); DIS_HIGH(G-BMULT5)

BO = ,BO OR ANDF((BMULT+24), (RMO+24));
!      G-B00 = AND(FF-RM06); DIS_HIGH(G-BMULT6)

BO = ,BO OR ANDF((BMULT+28), (RMO+28));
!      G-B00 = AND(FF-RM07); DIS_HIGH(G-BMULT7)

BO = ,BO OR ANDF((BMULT+32), (RMO+32));
!      G-B00 = AND(FF-RM08); DIS_HIGH(G-BMULT8)

BO = ,BO OR ANDF((BMULT+36), (RMO+36));
!      G-B00 = AND(FF-RM09); DIS_HIGH(G-BMULT9)

BO = ,BO OR ANDF((BMULT+40), (RMO+40));
!      G-B00 = AND(FF-RM010); DIS_HIGH(G-BMULT10)

BO = ,BO OR ANDF((BMULT+44), (RMO+44));
!      G-B00 = AND(FF-RM011); DIS_HIGH(G-BMULT11)

BO = ,BO OR ANDF((BMULT+48), (RMO+48));
!      G-B00 = AND(FF-RM012); DIS_HIGH(G-BMULT12)

BO = ,BO OR ANDF((BMULT+52), (RMO+52));
!      G-B00 = AND(FF-RM013); DIS_HIGH(G-BMULT13)

BO = ,BO OR ANDF((BMULT+56), (RMO+56));
!      G-B00 = AND(FF-RM014); DIS_HIGH(G-BMULT14)

BO = ,BO OR ANDF((BMULT+60), (RMO+60));
!      G-B00 = AND(FF-RM015); DIS_HIGH(G-BMULT15)

BO+4 = ANDF((BMULT), (RMO+64));
!      G-B01 = AND(FF-RM016); DIS_HIGH(G-BMULT0)

BO+4 = ,(BO+4) OR ANDF((BMULT+4), (RMO+68));

```

```

!      G-B01 = AND(FF-RM017); DIS_HIGH(G-BMULT1)

BO+4 = .(BO+4) OR ANDF((BMULT+8), (RM0+72));

!      G-B01 = AND(FF-RM018); DIS_HIGH(G-BMULT2)

BO+4 = .(BO+4) OR ANDF((BMULT+12), (RM0+76));

!      G-B01 = AND(FF-RM019); DIS_HIGH(G-BMULT3)

BO+4 = .(BO+4) OR ANDF((BMULT+16), (RM0+80));

!      G-B01 = AND(FF-RM020); DIS_HIGH(G-BMULT4)

BO+4 = .(BO+4) OR ANDF((BMULT+20), (RM0+84));

!      G-B01 = AND(FF-RM021); DIS_HIGH(G-BMULT5)

BO+4 = .(BO+4) OR ANDF((BMULT+24), (RM0+88));

!      G-B01 = AND(FF-RM022); DIS_HIGH(G-BMULT6)

BO+4 = .(BO+4) OR ANDF((BMULT+28), (RM0+92));

!      G-B01 = AND(FF-RM023); DIS_HIGH(G-BMULT7)

BO+4 = .(BO+4) OR ANDF((BMULT+32), (RM0+96));

!      G-B01 = AND(FF-RM024); DIS_HIGH(G-BMULT8)

BO+4 = .(BO+4) OR ANDF((BMULT+36), (RM0+100));

!      G-B01 = AND(FF-RM025); DIS_HIGH(G-BMULT9)

BO+4 = .(BO+4) OR ANDF((BMULT+40), (RM0+104));

!      G-B01 = AND(FF-RM026); DIS_HIGH(G-BMULT10)

BO+4 = .(BO+4) OR ANDF((BMULT+44), (RM0+108));

!      G-B01 = AND(FF-RM027); DIS_HIGH(G-BMULT11)

BO+4 = .(BO+4) OR ANDF((BMULT+48), (RM0+112));

!      G-B01 = AND(FF-RM028); DIS_HIGH(G-BMULT12)

BO+4 = .(BO+4) OR ANDF((BMULT+52), (RM0+116));

!      G-B01 = AND(FF-RM029); DIS_HIGH(G-BMULT13)

```

```

BO+4 = .(BO+4) OR ANDF((BMULT+56), (RMO+120));
!      G-BO1 = AND(FF-RMO30); DIS_HIGH(G-BMULT14)

BO+4 = .(BO+4) OR ANDF((BMULT+60), (RMO+124));
!      G-BO1 = AND(FF-RMO31); DIS_HIGH(G-BMULT15)

BO+8 = ANDF((BMULT), (RMO+128));
!      G-BO2 = AND(FF-RMO32); DIS_HIGH(G-BMULT0)

BO+8 = .(BO+8) OR ANDF((BMULT+4), (RMO+132));
!      G-BO2 = AND(FF-RMO33); DIS_HIGH(G-BMULT1)

BO+8 = .(BO+8) OR ANDF((BMULT+8), (RMO+136));
!      G-BO2 = AND(FF-RMO34); DIS_HIGH(G-BMULT2)

BO+8 = .(BO+8) OR ANDF((BMULT+12), (RMO+140));
!      G-BO2 = AND(FF-RMO35); DIS_HIGH(G-BMULT3)

BO+8 = .(BO+8) OR ANDF((BMULT+16), (RMO+144));
!      G-BO2 = AND(FF-RMO36); DIS_HIGH(G-BMULT4)

BO+8 = .(BO+8) OR ANDF((BMULT+20), (RMO+148));
!      G-BO2 = AND(FF-RMO37); DIS_HIGH(G-BMULT5)

BO+8 = .(BO+8) OR ANDF((BMULT+24), (RMO+152));
!      G-BO2 = AND(FF-RMO38); DIS_HIGH(G-BMULT6)

BO+8 = .(BO+8) OR ANDF((BMULT+28), (RMO+156));
!      G-BO2 = AND(FF-RMO39); DIS_HIGH(G-BMULT7)

BO+8 = .(BO+8) OR ANDF((BMULT+32), (RMO+160));
!      G-BO2 = AND(FF-RMO40); DIS_HIGH(G-BMULT8)

BO+8 = .(BO+8) OR ANDF((BMULT+36), (RMO+164));
!      G-BO2 = AND(FF-RMO41); DIS_HIGH(G-BMULT9)

```

TABLE 46 (CONT'D)

```

BO+8 = ,(BO+8) OR ANDF((BMULT+40), (RMO+168));
!      G-BO2 = AND(FF-RMO42); DIS_HIGH(G-BMULT10)

BO+8 = ,(BO+8) OR ANDF((BMULT+44), (RMO+172));
!      G-BO2 = AND(FF-RMO43); DIS_HIGH(G-BMULT11)

BO+8 = ,(BO+8) OR ANDF((BMULT+48), (RMO+176));
!      G-BO2 = AND(FF-RMO44); DIS_HIGH(G-BMULT12)

BO+8 = ,(BO+8) OR ANDF((BMULT+52), (RMO+180));
!      G-BO2 = AND(FF-RMO45); DIS_HIGH(G-BMULT13)

BO+8 = ,(BO+8) OR ANDF((BMULT+56), (RMO+184));
!      G-BO2 = AND(FF-RMO46); DIS_HIGH(G-BMULT14)

BO+8 = ,(BO+8) OR ANDF((BMULT+60), (RMO+188));
!      G-BO2 = AND(FF-RMO47); DIS_HIGH(G-BMULT15)

BO+12 =  ANDF((BMULT), (RMO+192));
!      G-BO3 = AND(FF-RMO48); DIS_HIGH(G-BMULT0)

BO+12 = ,(BO+12) OR ANDF((BMULT+4), (RMO+196));
!      G-BO3 = AND(FF-RMO49); DIS_HIGH(G-BMULT1)

BO+12 = ,(BO+12) OR ANDF((BMULT+8), (RMO+200));
!      G-BO3 = AND(FF-RMO50); DIS_HIGH(G-BMULT2)

BO+12 = ,(BO+12) OR ANDF((BMULT+12), (RMO+204));
!      G-BO3 = AND(FF-RMO51); DIS_HIGH(G-BMULT3)

BO+12 = ,(BO+12) OR ANDF((BMULT+16), (RMO+208));
!      G-BO3 = AND(FF-RMO52); DIS_HIGH(G-BMULT4)

BO+12 = ,(BO+12) OR ANDF((BMULT+20), (RMO+212));
!      G-BO3 = AND(FF-RMO53); DIS_HIGH(G-BMULT5)

```



```

BO+12 = ,(BO+12) OR ANDF((BMULT+24), (RMO+216));
!      G-BO3 = AND(FF-RMO54); DIS_HIGH(G-BMULT6)

BO+12 = ,(BO+12) OR ANDF((BMULT+28), (RMO+220));
!      G-BO3 = AND(FF-RMO55); DIS_HIGH(G-BMULT7)

BO+12 = ,(BO+12) OR ANDF((BMULT+32), (RMO+224));
!      G-BO3 = AND(FF-RMO56); DIS_HIGH(G-BMULT8)

BO+12 = ,(BO+12) OR ANDF((BMULT+36), (RMO+228));
!      G-BO3 = AND(FF-RMO57); DIS_HIGH(G-BMULT9)

BO+12 = ,(BO+12) OR ANDF((BMULT+40), (RMO+232));
!      G-BO3 = AND(FF-RMO58); DIS_HIGH(G-BMULT10)

BO+12 = ,(BO+12) OR ANDF((BMULT+44), (RMO+236));
!      G-BO3 = AND(FF-RMO59); DIS_HIGH(G-BMULT11)

BO+12 = ,(BO+12) OR ANDF((BMULT+48), (RMO+240));
!      G-BO3 = AND(FF-RMO60); DIS_HIGH(G-BMULT12)

BO+12 = ,(BO+12) OR ANDF((BMULT+52), (RMO+244));
!      G-BO3 = AND(FF-RMO61); DIS_HIGH(G-BMULT13)

BO+12 = ,(BO+12) OR ANDF((BMULT+56), (RMO+248));
!      G-BO3 = AND(FF-RMO62); DIS_HIGH(G-BMULT14)

BO+12 = ,(BO+12) OR ANDF((BMULT+60), (RMO+252));
!      G-BO3 = AND(FF-RMO63); DIS_HIGH(G-BMULT15)
!
!
!
!

```

! OUTPUT LATCHES FROM RAM SUBROUTINE

```

AOL = (.AOL AND .CLBB) OR (.AOL AND NOT .CLBB);
INV(AOL, AOL);

```

```

!      G-LA0 = AND(G-AOL, G-CLBB)
!      G-LB0 = AND(G-AOL, NOT(G-CLBB))
!      G-AOL = OR(G-LA0, G-LB0)

```

```

!           G-A0B = INV(G-A0L)

A1L = (.A0[1] AND .CLBB) OR (.A1L AND NOT .CLBB);
INV(A1B, A1L);

!           G-LA1 = AND(G-A01, G-CLBB)
!           G-LB1 = AND(G-A1L, NOT(G-CLBB))
!           G-A1L = OR(G-LA1, G-LB1)
!           G-A1B = INV(G-A1L)

A2L = (.A0[2] AND .CLBB) OR (.A2L AND NOT .CLBB);
INV(A2B, A2L);

!           G-LA2 = AND(G-A02, G-CLBB)
!           G-LB2 = AND(G-A2L, NOT(G-CLBB))
!           G-A2L = OR(G-LA2, G-LB2)
!           G-A2B = INV(G-A2L)

A3L = (.A0[3] AND .CLBB) OR (.A3L AND NOT .CLBB);
INV(A3B, A3L);

!           G-LA3 = AND(G-A03, G-CLBB)
!           G-LB3 = AND(G-A3L, NOT(G-CLBB))
!           G-A3L = OR(G-LA3, G-LB3)
!           G-A3B = INV(G-A3L)

B0L = (.B0[0] AND .CLBB) OR (.B0L AND NOT .CLBB);

!           G-LA0 = AND(G-B00, G-CLBB)
!           G-LB0 = AND(G-B0L, NOT(G-CLBB))
!           G-B0L = OR(G-LA0, G-LB0)

B1L = (.B0[1] AND .CLBB) OR (.B1L AND NOT .CLBB);

!           G-LA1 = AND(G-B01, G-CLBB)
!           G-LB1 = AND(G-B1L, NOT(G-CLBB))
!           G-B1L = OR(G-LA1, G-LB1)

B2L = (.B0[2] AND .CLBB) OR (.B2L AND NOT .CLBB);

!           G-LA2 = AND(G-B02, G-CLBB)
!           G-LB2 = AND(G-B2L, NOT(G-CLBB))
!           G-B2L = OR(G-LA2, G-LB2)

B3L = (.B0[3] AND .CLBB) OR (.B3L AND NOT .CLBB);

!           G-LA3 = AND(G-B03, G-CLBB)
!           G-LB3 = AND(G-B3L, NOT(G-CLBB))
!           G-B3L = OR(G-LA3, G-LB3)

```

!

! ALU INPUT MUX - SELECT SOURCES R AND S - CONTROL LOGIC

```

!      INV(I0B,.I0); INV(I1B,.I1); INV(I2B,.I2);
!      G-I0B = NOT(P-12)
!      G-I1B = NOT(P-13)
!      G-I2B = NOT(P-14)

```

```

!      INV(I2BB,I2B); NAND(ID,I0B,I1B);
!      G-I2BB = NOT(G-I2B)
!      G-ID = NAND(G-I0B, G-I1B)

```

! MUX LOGIC - ALU INPUT MUX

ROB = (NOT(AND3(I2B, I1B, A0L) OR AND3(ID, I2BB, .D0)));

```

!      G-AROL = AND(G-I2B, G-I1B, G-A0L)
!      G-AROR = AND(G-ID, G-I2BB, P-25)
!      G-ROB = NOR(G-AROL, G-AROR)

```

A = NOT .I0B; B = NANDF(I2BB, I1B);
SOB = (NOT(AND3(I2BB, I1B, A0L) OR AND3(A, I2B, B0L)
OR AND3(B, I0B, Q0L)));

```

!      G-ASOL = AND(G-I2BB, G-I1B, G-A0L)
!      G-ASOM = AND(INV(G-I0B), G-I2B, G-B0L)
!      G-ASON = OR(INV(G-I1B), INV(G-I2BB))
!      G-ASOR = AND(G-ASON, G-I0B, FF-Q0)
!      G-SOB = NOR(G-ASOL, G-ASOM, G-ASOR)

```

R1B = (NOT(AND3(I2B, I1B, A1L) OR AND3(ID, I2BB, .D1)));

```

!      G-AR1L = AND(G-I2B, G-I1B, G-A1L)
!      G-AR1R = AND(G-ID, G-I2BB, P-24)
!      G-R1B = NOR(G-AR1L, G-AR1R)

```

A = NOT .I0B; B = NANDF(I2BB, I1B);
S1B = (NOT(AND3(I2BB, I1B, A1L) OR AND3(A, I2B, B1L)
OR AND3(B, I0B, Q1L)));

```

!      G-AS1L = AND(G-I2BB, G-I1B, G-A1L)
!      G-AS1M = AND(INV(G-I0B), G-I2B, G-B1L)
!      G-AS1N = OR(INV(G-I1B), INV(G-I2BB))
!      G-AS1R = AND(G-AS1N, G-I0B, FF-Q1)
!      G-S1B = NOR(G-AS1L, G-AS1M, G-AS1R)

```

R2B = (NOT(AND3(I2B, I1B, A2L) OR AND3(ID, I2BB, .D2)));

```

!      G-AR2L = AND(G-I2B, G-I1B, G-A2L)
!      G-AR2R = AND(G-ID, G-I2BB, P-23)
!      G-R2B = NOR(G-AR2L, G-AR2R)

```

BLE 46 (CONT'D)

A = NOT .I0B; B = NANDF(I2BB, I1B);
S2B = (NOT(AND3(I2BB, I1B, A2L) OR AND3(A, I2B, B2L)

```

OR AND3(B, 10B, Q2L));

!      G-AS2L = AND(G-I2BB, G-I1B, G-A2L)
!      G-AS2M = AND(INV(G-I0B), G-I2B, G-B2L)
!      G-AS2N = OR(INV(G-I1B), INV(G-I2BB))
!      G-AS2R = AND(G-AS2N, G-I0B, FF-Q2)
!      G-S2B = NOR(G-AS2L, G-AS2M, G-AS2R)

R3B = (NOT(AND3(I2B, I1B, A3L) OR AND3(ID, I2BB, .D3)));

!      G-AR3L = AND(G-I2B, G-I1B, G-A3L)
!      G-AR3R = AND(G-ID, G-I2BB, P-22)
!      G-R3B = NOR(G-AR3L, G-AR3R)

A = NOT .I0B;   B = NAND(I2BB, I1B);
S3B = (NOT(AND3(I2BB, I1B, A3L) OR AND3(A, I2B, B3L)
OR AND3(B, 10B, Q3L)));

!      G-AS3L = AND(G-I2BB, G-I1B, G-A3L)
!      G-AS3M = AND(INV(G-I0B), G-I2B, G-B3L)
!      G-AS3N = OR(INV(G-I1B), INV(G-I2BB))
!      G-AS3R = AND(G-AS3N, G-I0B, FF-Q3)
!      G-S3B = NOR(G-AS3L, G-AS3M, G-AS3R)

! ALU CONTROL LOGIC

INV(CR,.I3);INV(CS,.I4);INV(FC,.I5);

!      G-CR = NOT(P-26)
!      G-CS = NOT(P-28)
!      G-FC = NOT(P-27)

INH = (NOT(AND3(.I3, .I4, FC) OR AND(.I5, CS)));

!      G-INF = AND(G-CS, P-27)
!      G-ING = AND(G-FC, P-28, P-26)
!      G-INH = NOR(G-INF, G-ING)

! ALU FIRST LEVEL - GENERATE P AND G SIGNALS

A = NOT (EXOR(CR, R0B));
!      G-R0B1 = XNOR(G-CR, G-R0B)

B = NOT (EXOR(CS, S0B));
!      G-S0B1 = XNOR(G-CS, G-S0B)

NAND3(P0, INH, A, B);
!      G-P0 = NAND(G-INH, G-R0B1, G-S0B1)

NOR(G0, A, B);
!      G-G0 = NOR(G-R0B1, G-S0B1)

A = NOT (EXOR(CR, R1B));
!      G-R1B1 = XNOR(G-CR, G-R1B)

B = NOT (EXOR(CS, S1B));
!      G-S1B1 = XNOR(G-CS, G-S1B)

```

```

!      NAND3(P1, INH, A, B);
!      G-P1 = NAND(G-INH, G-R1B1, G-S1B1)

!      NOR(G1, A, B);
!      G-G1 = NOR(G-R1B1, G-S1B1)

!      A = NOT (EXOR(CR, R2B));
!      G-R2B1 = XNOR(G-CR, G-R2B)

!      B = NOT (EXOR(CS, S2B));
!      G-S2B1 = XNOR(G-CS, G-S2B)

!      NAND3(P2, INH, A, B);
!      G-P2 = NAND(G-INH, G-R2B1, G-S2B1)

!      NOR(G2, A, B);
!      G-G2 = NOR(G-R2B1, G-S2B1)

!      A = NOT (EXOR(CR, R3B));
!      G-R3B1 = XNOR(G-CR, G-R3B)

!      B = NOT (EXOR(CS, S3B));
!      G-S3B1 = XNOR(G-CS, G-S3B)

!      NAND3(P3, INH, A, B);
!      G-P3 = NAND(G-INH, G-R3B1, G-S3B1)

!      NOR(G3, A, B);
!      G-G3 = NOR(G-R3B1, G-S3B1)

!      ! ALU SECOND LEVEL - GENERATE CARRY SIGNALS

!      NAND(CAR0, .CN, INH);
!      G-CN0 = AND(G-INH, P-29)
!      G-CAR0 = NOR(G-CN0)

!      CAR1 = (NOT(AND3(INH, P0, .CN) OR ANDF(INH, G0)));

!      G-CN1A = AND(G-INH, G-G0)
!      G-CN1B = AND(INH, G-P0, P-29)
!      G-CAR1 = NOR(CN1A, CN1B)

!      AND4(A, .CN, P0, P1, INH);
!      CAR2 = (NOT(.A OR AND3(G0, INH, P1) OR ANDF(G1, INH)));

!      G-CN2A = AND(G-INH, G-G1)
!      G-CN2B = AND(G-P1, G-INH, G-G0)
!      G-CN2C = AND(G-INH, G-P1, G-P0, P-29)
!      G-CAR2 = NOR(G-CN2A, G-CN2B, G-CN2C)

!      NAND5(A, .CN, P0, P1, P2, INH);
!      NAND4(B, G0, P1, P2, INH);
!      CAR3 = (NOT((NOT .A) OR (NOT .B) OR AND3(G1, P2, INH)
!      OR ANDF(G2, INH)));

!      G-CN3A = AND(G-INH, G-G2)
!      G-CN3B = AND(G-INH, G-P2, G-G1)

```

```

!      G-CN3C = AND(G-INH, G-P2, G-P1, G-G0)
!      G-CN3D = AND(G-INH, G-P2, G-P1, G-P0, P-29)
!      G-CAR3 = NOR(G-CN3A, G-CN3B, G-CN3C, G-CN3D)
!

```

! ALU - GENERATE INTERCHIP CARRY SIGNALS

```

NAND4(A, G0, P1, P2, P3);
  BUF(B, G3);
.GBAR = (NOT((NOT .A) OR AND3(G1, P2, P3) OR ANDF(G2, P3) OR

!      G-GB1 = AND(G-G0, G-P1, G-P2, G-P3)
!      G-GB2 = AND(G-G1, G-P2, G-P3)
!      G-GB3 = AND(G-G2, G-P3)
!      G-GB4 = AND(G-G3)
!      G-GBAR(P-32) = NOR(G-GB1, G-GB2, G-GB3, G-GB4)

NAND5(TEMP, .CN, P3, P2, P1, P0);
NAND4(.PBAR, P3, P2, P1, P0);
!      G-PB1 = NAND(P-29, G-P3, G-P2, G-P1, G-P0)
!      G-PBAR(P-35) = NAND(G-P3, G-P2, G-P1, G-P0)

NAND(.CN4, TEMP, .GBAR);
ANDR(TEMP, TEMP, .GBAR);
XORR(.OVR, TEMP, CAR3);

!      G-CN4(P-33) = NAND(G-GBAR, G-PB1)
!      G-PB3 = AND(G-GBAR, G-PB1)
!      G-OVR(P-34) = XOR(G-PB3, G-CAR3)
!

```

! ALU OUTPUT BITS AND THEIR INVERSES

```

A = NOT .G0;
F0 = NOT (NANDF(P0, A) XOR ANDF(CAR0, FC));

!      G-PG0 = NAND(G-P0, INV(G-G0))
!      G-CARA = AND(G-FC, G-CAR0)
!      G-F0 = XNOR(G-CARA, G-PG0)

A = NOT .G1;
F1 = NOT (NANDF(P1, A) XOR ANDF(CAR1, FC));

!      G-PG1 = NAND(G-P1, INV(G-G1))
!      G-CARB = AND(G-FC, G-CAR1)
!      G-F1 = XNOR(G-CARB, G-PG1)

A = NOT .G2;
F2 = NOT (NANDF(P2, A) XOR ANDF(CAR2, FC));

!      G-PG2 = NAND(G-P2, INV(G-G2))
!      G-CARC = AND(G-FC, G-CAR2)
!      G-F2 = XNOR(G-CARC, G-PG2)

A = NOT .G3;
F3 = NOT (NANDF(P3, A) XOR ANDF(CAR3, FC));

!      G-PG3 = NAND(G-P3, INV(G-G3))
!      G-CARD = AND(G-FC, G-CAR3)
!      G-F3 = XNOR(G-CARD, G-PG3)
!

```

```

      INV(F0B,F0);
!      G-F0B = NOT(G-F0)

      INV(F1B,F1);
!      G-F1B = NOT(G-F1)

      INV(F2B,F2);
!      G-F2B = NOT(G-F2)

      INV(F3B,F3);
!      G-F3B(P-31) = NOT(G-F3)
!

      ! ALU MAGNITUDE CHECKER OUTPUTS

      .F3P=.F3B;
      NOR4(TEMP,F0B,F1B,F2B,F3B);
      .FEU=..FEU AND .TEMP;
!      G-FE0(P-11) = NOR(G-F0B, G-F1B, G-F2B, G-F3B)
!

      ! DESTINATION CONTROL LOGIC

      INV(I6B,.I6); INV(I7B,.I7); INV(NS,.I8);
!      G-I6B = NOT(P-5)
!      G-I7B = NOT(P-6)
!      G-NS = NOT(P-7)

      INV(I6BB,I6B); NOR(SD,.I7,NS);
!      G-I6BB = NOT(G-I6B)
!      G-SD = NOR(P-6, G-NS)

      NOR(SU,NS,I7B); NAND(RE,NS,I7B);
!      G-SU = NOR(G-NS, G-I7B)
!      G-RE = NAND(G-I7B, G-NS)

      ANDR(TEMP,RE,NS); NOR(QE,I6BB,TEMP);
!      G-GT = AND(G-RE, G-NS)
!      G-QE = NOR(G-I6BB, G-GT)

      NAND3(SF,RE,I6B,NS);
!      G-SF = NAND(G-I6B, G-RE, G-NS)

      ANDR(WE,RE,CLBB);
!      G-WE = AND(G-RE, INV(G-CLBB))

      TEMP = NOT .SD;
      SFB = NOT .SU;

      ORR(.Q0,Q0L,TEMP);
!      GTS-Q0(P-21) = NOT(FF-Q0');
!      DIS_LOW(G-SD)

      ORR(.Q3,Q3L,SFB);
!      GTS-Q3(P-16) = NOT(FF-Q3');
!      DIS_LOW(G-SU)

      ORR(.RAM0,F0B,TEMP);
!      GTS-RAM0(P-9) = NOT(G-F0B);

```

```

!                                     DIS_LOW(G-SD)

ORR(,RAM3,F3B,SFB);
!                                     GTS-RAM3(P-8) = NOT(G-F3B);
!                                     DIS_LOW(G-SU)
!

! CHIP DATA OUTPUT

INV(SFB,SF);
!                                     G-SFB = NOT(G-SF)

INV(CE,,CEBAR);
!                                     G-CE = NOT(P-40)

B = (NOT(ANDF(SF, F0) OR ANDF(SFB, A0B)));
TEMP = NOT .CE;
ORR (.Y0, B, TEMP);

!                                     G-Y0A = AND(G-SF, G-F0)
!                                     G-Y0B = AND(G-SFB, G-A0B)
!                                     G-Y0(P-36) = NOR(G-Y0A, G-Y0B); DIS_LOW(G-CE)

B = (NOT(ANDF(SF, F1) OR ANDF(SFB, A1B)));
TEMP = NOT .CE;
ORR (.Y1, B, TEMP);

!                                     G-Y1A = AND(G-SF, G-F1)
!                                     G-Y1B = AND(G-SFB, G-A1B)
!                                     G-Y1(P-37) = NOR(G-Y1A, G-Y1B); DIS_LOW(G-CE)

B = (NOT(ANDF(SF, F2) OR ANDF(SFB, A2B)));
TEMP = NOT .CE;
ORR (.Y2, B, TEMP);

!                                     G-Y2A = AND(G-SF, G-F2)
!                                     G-Y2B = AND(G-SFB, G-A2B)
!                                     G-Y2(P-38) = NOR(G-Y2A, G-Y2B); DIS_LOW(G-CE)

B = (NOT(ANDF(SF, F3) OR ANDF(SFB, A3B)));
TEMP = NOT .CE;
ORR (.Y3, B, TEMP);

!                                     G-Y3A = AND(G-SF, G-F3)
!                                     G-Y3B = AND(G-SFB, G-A3B)
!                                     G-Y3(P-39) = NOR(G-Y3A, G-Y3B); DIS_LOW(G-CE)
!
END;
```

GLOBAL ROUTINE TALUOUT(Q0,Q3,RAM0,RAM3,K,JJ) : novalue =
BEGIN

```

require 'BINDM.R32';
LOCAL T0,T1,T2,T3,A,B;
LOCAL GATE;
```


! Q REGISTER SELECT LOGIC

T0 = (ANDF(SU, .Q0) OR ANDF(NS, F0B) OR ANDF(SD, Q1L));

```
!           G-QSU0 = AND(G-SU, GTS-Q0)
!           G-QNS0 = AND(G-NS, G-F0B)
!           G-QSD0 = AND(G-SD, FF-Q1)
!           G-QT0  = NOR(G-QSU0, G-QNS0, G-QSD0)
```

T1 = (ANDF(SU, Q0L) OR ANDF(NS, F1B) OR ANDF(SD, Q2L));

```
!           G-QSU1 = AND(G-SU, FF-Q0)
!           G-QNS1 = AND(G-NS, G-F1B)
!           G-QSD1 = AND(G-SD, FF-Q2)
!           G-QT1  = NOR(G-QSU1, G-QNS1, G-QSD1)
```

T2 = (ANDF(SU, Q1L) OR ANDF(NS, F2B) OR ANDF(SD, Q3L));

```
!           G-QSU2 = AND(G-SU, FF-Q1)
!           G-QNS2 = AND(G-NS, G-F2B)
!           G-QSD2 = AND(G-SD, FF-Q3)
!           G-QT2  = NOR(G-QSU2, G-QNS2, G-QSD2)
```

T3 = (ANDF(SU, Q2L) OR ANDF(NS, F3B) OR ANDF(SD, .Q3));

```
!           G-QSU3 = AND(G-SU, FF-Q2)
!           G-QNS3 = AND(G-NS, G-F3B)
!           G-QSD3 = AND(G-SD, GTS-Q3)
!           G-QT3  = NOR(G-QSU3, G-QNS3, G-QSD3)
!
```

! Q REGISTER

```
!           A = NOT .CLB;  A = ANDF(A, QE);
!           G-QCLK = AND(INV(G-CLB), G-QE)
```

```
!           Q0L = (.T0 AND .A) OR (.Q0L AND NOT .A);
!           FF-Q0 = D(DATA=G-QT0,CLK_UP=G-QCLK)
```

INV(Q0B, Q0L);

```
!           Q1L = (.T1 AND .A) OR (.Q1L AND NOT .A);
!           FF-Q1 = D(DATA=G-QT1,CLK_UP=G-QCLK)
```

INV(Q1B, Q1L);

```

!      Q2L = (.T2 AND .A) OR (.Q2L AND NOT .A);
!      FF-Q2 = D(DATA=G-QT2,CLK_UP=G-QCLK)

      INV(Q2B, Q2L);

```

```

!      Q3L = (.T3 AND .A) OR (.Q3L AND NOT .A);
!      FF-Q3 = D(DATA=G-QT3,CLK_UP=G-QCLK)

      INV(Q3B, Q3L);
!

```

! RAM MULTIPLEXOR LOGIC

```

      T0 = (ANDF(SU, .RAM0) OR ANDF(NS, F0B) OR ANDF(SD, F1B));

!      G-TSU0 = AND(G-SU, GTS-RAM0)
!      G-TNS0 = AND(G-NS, G-F0B)
!      G-TSD0 = AND(G-SD, G-F1B)
!      G-TT0 = NOR(G-TSU0, G-TNS0, G-TSD0)

```

```

      T1 = (ANDF(SU, F0B) OR ANDF(NS, F1B) OR ANDF(SD, F2B));

!      G-TSU1 = AND(G-SU, G-F0B)
!      G-TNS1 = AND(G-NS, G-F1B)
!      G-TSD1 = AND(G-SD, G-F2B)
!      G-TT1 = NOR(G-TSU1, G-TNS1, G-TSD1)

```

```

      T2 = (ANDF(SU, F1B) OR ANDF(NS, F2B) OR ANDF(SD, F3B));

!      G-TSU2 = AND(G-SU, G-F1B)
!      G-TNS2 = AND(G-NS, G-F2B)
!      G-TSD2 = AND(G-SD, G-F3B)
!      G-TT2 = NOR(G-TSU2, G-TNS2, G-TSD2)

```

```

      T3 = (ANDF(SU, F2B) OR ANDF(NS, F3B) OR ANDF(SD, .RAM3));

!      G-TSU3 = AND(G-SU, G-F2B)
!      G-TNS3 = AND(G-NS, G-F3B)
!      G-TSD3 = AND(G-SD, GTS-RAM3)
!      G-TT3 = NOR(G-TSU3, G-TNS3, G-TSD3)
!
!

```

! CALL THE RAM SUBROUTINE

```

GATE = ANDF(BMULT, WE);
!      G-ATE1 = AND(G-BMULT0, G-WE)

```

```

RMO = (,RMO AND NOT ,GATE) OR (,T0 AND ,GATE);
!      FF-RM00 = D(DATA=G-IT0,CLK_UP=G-ATE1)

RMO+64 = (,(RMO+64) AND NOT ,GATE) OR (,T1 AND ,GATE);
!      FF-RM016 = D(DATA=G-TT1,CLK_UP=G-ATE1)

RMO+128 = (,(RMO+128) AND NOT ,GATE) OR (,T2 AND ,GATE);
!      FF-RM032 = D(DATA=G-TT2,CLK_UP=G-ATE1)

RMO+192 = (,(RMO+192) AND NOT ,GATE) OR (,T3 AND ,GATE);
!      FF-RM048 = D(DATA=G-TT3,CLK_UP=G-ATE1)

GATE = ANDF((BMULT+4), WE);
!      G-ATE2 = AND(G-BMULT1, G-WE)

RMO+4 = (,(RMO+4) AND NOT ,GATE) OR (,T0 AND ,GATE);
!      FF-RM01 = D(DATA=G-IT0,CLK_UP=G-ATE2)

RMO+68 = (,(RMO+68) AND NOT ,GATE) OR (,T1 AND ,GATE);
!      FF-RM017 = D(DATA=G-TT1,CLK_UP=G-ATE2)

RMO+132 = (,(RMO+132) AND NOT ,GATE) OR (,T2 AND ,GATE);
!      FF-RM033 = D(DATA=G-TT2,CLK_UP=G-ATE2)

RMO+196 = (,(RMO+196) AND NOT ,GATE) OR (,T3 AND ,GATE);
!      FF-RM049 = D(DATA=G-TT3,CLK_UP=G-ATE2)

GATE = ANDF((BMULT+8), WE);
!      G-ATE3 = AND(G-BMULT2, G-WE)

RMO+8 = (,(RMO+8) AND NOT ,GATE) OR (,T0 AND ,GATE);
!      FF-RM02 = D(DATA=G-IT0,CLK_UP=G-ATE3)

RMO+72 = (,(RMO+72) AND NOT ,GATE) OR (,T1 AND ,GATE);
!      FF-RM018 = D(DATA=G-TT1,CLK_UP=G-ATE3)

RMO+136 = (,(RMO+136) AND NOT ,GATE) OR (,T2 AND ,GATE);
!      FF-RM034 = D(DATA=G-TT2,CLK_UP=G-ATE3)

RMO+200 = (,(RMO+200) AND NOT ,GATE) OR (,T3 AND ,GATE);
!      FF-RM050 = D(DATA=G-TT3,CLK_UP=G-ATE3)

GATE = ANDF((BMULT+12), WE);
!      G-ATE4 = AND(G-BMULT3, G-WE)

```

```

RMO+12 = (.(RMO+12) AND NOT .GATE) OR (.(T0 AND .GATE));
!      FF-RMO3 = D(DATA=G-TT0,CLK_UP=G-ATE4)

RMO+76 = (.(RMO+76) AND NOT .GATE) OR (.(T1 AND .GATE));
!      FF-RMO19 = D(DATA=G-TT1,CLK_UP=G-ATE4)

RMO+140 = (.(RMO+140) AND NOT .GATE) OR (.(T2 AND .GATE));
!      FF-RMO35 = D(DATA=G-TT2,CLK_UP=G-ATE4)

RMO+204 = (.(RMO+204) AND NOT .GATE) OR (.(T3 AND .GATE));
!      FF-RMO51 = D(DATA=G-TT3,CLK_UP=G-ATE4)

GATE = ANDF((BMULT+16), WE);
!      G-ATE5 = AND(G-BMULT4, G-WE)

RMO+16 = (.(RMO+16) AND NOT .GATE) OR (.(T0 AND .GATE));
!      FF-RMO4 = D(DATA=G-TT0,CLK_UP=G-ATE5)

RMO+80 = (.(RMO+80) AND NOT .GATE) OR (.(T1 AND .GATE));
!      FF-RMO20 = D(DATA=G-TT1,CLK_UP=G-ATE5)

RMO+144 = (.(RMO+144) AND NOT .GATE) OR (.(T2 AND .GATE));
!      FF-RMO36 = D(DATA=G-TT2,CLK_UP=G-ATE5)

RMO+208 = (.(RMO+208) AND NOT .GATE) OR (.(T3 AND .GATE));
!      FF-RMO52 = D(DATA=G-TT3,CLK_UP=G-ATE5)

GATE = ANDF((BMULT+20), WE);
!      G-ATE6 = AND(G-BMULT5, G-WE)

RMO+20 = (.(RMO+20) AND NOT .GATE) OR (.(T0 AND .GATE));
!      FF-RMO5 = D(DATA=G-TT0,CLK_UP=G-ATE6)

RMO+84 = (.(RMO+84) AND NOT .GATE) OR (.(T1 AND .GATE));
!      FF-RMO21 = D(DATA=G-TT1,CLK_UP=G-ATE6)

RMO+148 = (.(RMO+148) AND NOT .GATE) OR (.(T2 AND .GATE));
!      FF-RMO37 = D(DATA=G-TT2,CLK_UP=G-ATE6)

RMO+212 = (.(RMO+212) AND NOT .GATE) OR (.(T3 AND .GATE));
!      FF-RMO53 = D(DATA=G-TT3,CLK_UP=G-ATE6)

GATE = ANDF((BMULT+24), WE);
!      G-ATE7 = AND(G-BMULT6, G-WE)

```

```

!      FF-RM06 = D(DATA=G-TT0,CLK_UP=G-ATE7)

RMO+88 = (.(RMO+88) AND NOT .GATE) OR (.(T1 AND .GATE));
!      FF-RM022 = D(DATA=G-TT1,CLK_UP=G-ATE7)

RMO+152 = (.(RMO+152) AND NOT .GATE) OR (.(T2 AND .GATE));
!      FF-RM038 = D(DATA=G-TT2,CLK_UP=G-ATE7)

RMO+216 = (.(RMO+216) AND NOT .GATE) OR (.(T3 AND .GATE));
!      FF-RM054 = D(DATA=G-TT3,CLK_UP=G-ATE7)

GATE = ANDF((BMULT+28), WE);
!      G-ATE8 = AND(G-BMULT7, G-WE)

RMO+28 = (.(RMO+28) AND NOT .GATE) OR (.(T0 AND .GATE));
!      FF-RM07 = D(DATA=G-TT0,CLK_UP=G-ATE8)

RMO+92 = (.(RMO+92) AND NOT .GATE) OR (.(T1 AND .GATE));
!      FF-RM023 = D(DATA=G-TT1,CLK_UP=G-ATE8)

RMO+156 = (.(RMO+156) AND NOT .GATE) OR (.(T2 AND .GATE));
!      FF-RM039 = D(DATA=G-TT2,CLK_UP=G-ATE8)

RMO+220 = (.(RMO+220) AND NOT .GATE) OR (.(T3 AND .GATE));
!      FF-RM055 = D(DATA=G-TT3,CLK_UP=G-ATE8)

GATE = ANDF((BMULT+32), WE);
!      G-ATE9 = AND(G-BMULT8, G-WE)

RMO+32 = (.(RMO+32) AND NOT .GATE) OR (.(T0 AND .GATE));
!      FF-RM08 = D(DATA=G-TT0,CLK_UP=G-ATE9)

RMO+96 = (.(RMO+96) AND NOT .GATE) OR (.(T1 AND .GATE));
!      FF-RM024 = D(DATA=G-TT1,CLK_UP=G-ATE9)

RMO+160 = (.(RMO+160) AND NOT .GATE) OR (.(T2 AND .GATE));
!      FF-RM040 = D(DATA=G-TT2,CLK_UP=G-ATE9)

RMO+224 = (.(RMO+224) AND NOT .GATE) OR (.(T3 AND .GATE));
!      FF-RM056 = D(DATA=G-TT3,CLK_UP=G-ATE9)

GATE = ANDF((BMULT+36), WE);
!      G-ATE10 = AND(G-BMULT9, G-WE)

RMO+36 = (.(RMO+36) AND NOT .GATE) OR (.(T0 AND .GATE));
!      FF-RM09 = D(DATA=G-TT0,CLK_UP=G-ATE10)

```

```

RMO+100 = (.(RMO+100) AND NOT .GATE) OR (.(T1 AND .GATE);
!      FF-RMO25 = D(DATA=G-TT1,CLK_UP=G-ATE10)

RMO+164 = (.(RMO+164) AND NOT .GATE) OR (.(T2 AND .GATE);
!      FF-RMO41 = D(DATA=G-TT2,CLK_UP=G-ATE10)

RMO+228 = (.(RMO+228) AND NOT .GATE) OR (.(T3 AND .GATE);
!      FF-RMO57 = D(DATA=G-TT3,CLK_UP=G-ATE10)

GATE = ANDF((BMULT+40), WE);
!      G-ATE11 = AND(G-BMULT10, G-WE)

RMO+40 = (.(RMO+40) AND NOT .GATE) OR (.(T0 AND .GATE);
!      FF-RMO10 = D(DAIA=G-TT0,CLK_UP=G-ATE11)

RMO+104 = (.(RMO+104) AND NOT .GATE) OR (.(T1 AND .GATE);
!      FF-RMO26 = D(DATA=G-TT1,CLK_UP=G-ATE11)

RMO+168 = (.(RMO+168) AND NOT .GATE) OR (.(T2 AND .GATE);
!      FF-RMO42 = D(DATA=G-TT2,CLK_UP=G-ATE11)

RMO+232 = (.(RMO+232) AND NOT .GATE) OR (.(T3 AND .GATE);
!      FF-RMO58 = D(DATA=G-TT3,CLK_UP=G-ATE11)

GATE = ANDF((BMULT+44), WE);
!      G-ATE12 = AND(G-BMULT11, G-WE)

RMO+44 = (.(RMO+44) AND NOT .GATE) OR (.(T0 AND .GATE);
!      FF-RMO11 = D(DATA=G-TT0,CLK_UP=G-ATE12)

RMO+108 = (.(RMO+108) AND NOT .GATE) OR (.(T1 AND .GATE);
!      FF-RMO27 = D(DATA=G-TT1,CLK_UP=G-ATE12)

RMO+172 = (.(RMO+172) AND NOT .GATE) OR (.(T2 AND .GATE);
!      FF-RMO43 = D(DATA=G-TT2,CLK_UP=G-ATE12)

RMO+236 = (.(RMO+236) AND NOT .GATE) OR (.(T3 AND .GATE);
!      FF-RMO59 = D(DATA=G-TT3,CLK_UP=G-ATE12)

GATE = ANDF((BMULT+48), WE);
!      G-ATE13 = AND(G-BMULT12, G-WE)

RMO+48 = (.(RMO+48) AND NOT .GATE) OR (.(T0 AND .GATE);
!      FF-RMO12 = D(DATA=G-TT0,CLK_UP=G-ATE13)

```

TABLE 46
(CONT'D)

```

RMO+112 = (.(RMO+112) AND NOT .GATE) OR (.(T1 AND .GATE));
!      FF-RMU28 = D(DATA=G-TT1,CLK_UP=G-ATE13)

RMO+176 = (.(RMO+176) AND NOT .GATE) OR (.(T2 AND .GATE));
!      FF-RMU44 = D(DATA=G-TT2,CLK_UP=G-ATE13)

RMO+240 = (.(RMO+240) AND NOT .GATE) OR (.(T3 AND .GATE));
!      FF-RMU60 = D(DATA=G-TT3,CLK_UP=G-ATE13)

GATE = ANDF((BMULT+52), WE);
!      G-ATE14 = AND(G-BMULT13, G-WE)

RMO+52 = (.(RMO+52) AND NOT .GATE) OR (.(T0 AND .GATE));
!      FF-RMU13 = D(DATA=G-TT0,CLK_UP=G-ATE14)

RMO+116 = (.(RMO+116) AND NOT .GATE) OR (.(T1 AND .GATE));
!      FF-RMU29 = D(DATA=G-TT1,CLK_UP=G-ATE14)

RMO+180 = (.(RMO+180) AND NOT .GATE) OR (.(T2 AND .GATE));
!      FF-RMU45 = D(DATA=G-TT2,CLK_UP=G-ATE14)

RMO+244 = (.(RMO+244) AND NOT .GATE) OR (.(T3 AND .GATE));
!      FF-RMU61 = D(DATA=G-TT3,CLK_UP=G-ATE14)

GATE = ANDF((BMULT+56), WE);
!      G-ATE15 = AND(G-BMULT14, G-WE)

RMO+56 = (.(RMO+56) AND NOT .GATE) OR (.(T0 AND .GATE));
!      FF-RMU14 = D(DATA=G-TT0,CLK_UP=G-ATE15)

RMO+120 = (.(RMO+120) AND NOT .GATE) OR (.(T1 AND .GATE));
!      FF-RMU30 = D(DATA=G-TT1,CLK_UP=G-ATE15)

RMO+184 = (.(RMO+184) AND NOT .GATE) OR (.(T2 AND .GATE));
!      FF-RMU46 = D(DATA=G-TT2,CLK_UP=G-ATE15)

RMO+248 = (.(RMO+248) AND NOT .GATE) OR (.(T3 AND .GATE));
!      FF-RMU62 = D(DATA=G-TT3,CLK_UP=G-ATE15)

GATE = ANDF((BMULT+60), WE);
!      G-ATE16 = AND(G-BMULT15, G-WE)

RMO+60 = (.(RMO+60) AND NOT .GATE) OR (.(T0 AND .GATE));
!      FF-RMU15 = D(DATA=G-TT0,CLK_UP=G-ATE16)

```

RMO+124 = (.(RMO+124) AND NOT ,GATE) OR (,T1 AND ,GATE);
!
FF-RMO31 = D(DATA=G-TT1,CLK_UP=G-ATE16)

RMO+188 = (.(RMO+188) AND NOT ,GATE) OR (,T2 AND ,GATE);
!
FF-RMO47 = D(DATA=G-TT2,CLK_UP=G-ATE16)

RMO+252 = (.(RMO+252) AND NOT ,GATE) OR (,T3 AND ,GATE);
!
FF-RMO63 = D(DATA=G-TT3,CLK_UP=G-ATE16)

END;

!
!
!

S END CHIP S

!
!
!
NOTE: SOURCE DATA - ADVANCED MICRO DEVICES
"THE AM2900 FAMILY DATA BOOK"
AND ATTACHED DRAWINGS

END ELUDOM

APPENDIX C

TIMING AND CONTROL CARD CHIP LIBRARIES

IN EMULATION SYNTAX AND IN BLISS

\$ CHIP DEFINITION \$

TYPE: 74_LS_00
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD 2-INPUT POSITIVE-NAND GATES.
UNUSED PINS: NONE
FUNCTIONS:

G-01(P-3) = NAND(P-1, P-2)
G-02(P-6) = NAND(P-4, P-5)
G-03(P-8) = NAND(P-9, P-10)
G-04(P-11) = NAND(P-12, P-13)

\$ END CHIP \$

TABLE 47

\$ CHIP DEFINITION \$

TYPE: 74_LS_08
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD 2-INPUT POSITIVE-AND GATES.
UNUSED PINS: NONE

FUNCTIONS:

G-01(P-3) = AND(P-1, P-2,)
G-02(P-6) = AND(P-4, P-5)
G-03(P-8) = AND(P-9, P-10,)
G-04(P-11) = AND(P-12, P-13)

\$ END CHIP \$

\$ DEFINE CHIP \$

TYPE: 74_S_10

FAMILY: TTL

POWER: VCC = P14, GND = 7

DESCRIPTION: TRIPLE 3-INPUT POSITIVE NAND GATES.

UNUSED PINS: NONE

FUNCTIONS:

G-01(P-12) = AND(P-1, P-2, P-13)

G-02(P-6) = AND(P-3, P-4, P-5)

G-03(P-8) = AND(P-9, P-10, P-11)

\$ END CHIP \$

\$ CHIP DEFINITION \$

TYPE: 74-LS-11
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: TRIPLE 3-INPUT POSITIVE AND GATES.
UNUSED PINS: NONE
FUNCTIONS:

G-01(P-12) = AND(P-1, P-2, P-13)

G-02(P-6) = AND(P-3, P-4, P-5)

G-03(P-8) = AND(P-9, P-10, P-11)

\$ END CHIP \$

S CHIP DEFINITION S

TYPE: 74-S-20
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: DUAL 4-INPUT POSITIVE-NAND GATES.
UNUSED PINS: P-3, P-11

FUNCTIONS:

G-01(P-6) = NAND(P-1, P-2, P-4, P-5)

G-02(P-8) = NAND(P-9, P-10, P-12, P-13)

S END CHIP S

\$ CHIP DEFINITION \$

TYPE: 74_S_37
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: QUAD 2-INPUT POSITIVE-NAND BUFFERS.
UNUSED PINS: NONE

FUNCTIONS:

G-01(P-3) = NAND(P-1, P-2)
G-02(P-6) = NAND(P-4, P-5)
G-03(P-8) = NAND(P-9, P-10)
G-04(P-11) = NAND(P-12, P-13)

\$ END CHIP \$

TABLE 52

S CHIP DEFINITION S

TYPE: 74_40
FAMILY: TTL
POWER: VCC = P-14, GND = P-7
DESCRIPTION: DUAL 4-INPUT POSITIVE-NAND BUFFERS.
UNUSED PINS: P-3, P-11

FUNCTIONS:

G-01(P-6) = NAND(P-1, P-2, P-4, P-5)
G-02(P-8) = NAND(P-9, P-10, P-12, P-13)

S END CHIP S

APPENDIX D

BDX930 PROCESSOR DESCRIPTION

This Appendix contains:

- o Table of card labels and their respective position designations,
- o Card file interconnections, and
- o Description of each external connection pertaining to the BDX930 processor as emulated.

CARD ASSIGNMENT TABLE

J9	TIMING AND CONTROL
J10	CPU

TABLE 54

CARD INTERCONNECTION TABLE

SOURCE		FEEDS	
J9-5A	->	J10-44A	BBUF
J9-25A	->	J10-17A	QMI*
J9-30C	->	J10-29B	ECUT*
J10-3B	->	J09-15C	IND*
J10-24B	->	J10-8A	FLAG/SAT*
J10-10B	->	J10-45A	QMAR*
J10-41A	->	J09-18C	U29
J10-42A	->	J09-21C	U26
J10-43A	->	J09-24C	U28*
J10-44B	->	J09-25C	U30*
J10-47B	->	J09-21B	U27*
J10-48B	->	J09-23C	U27
J10-49B	->	J09-4C	U30

TABLE 55

COMPUTER CONNECTOR PINS

CONN PIN	SIG IDENT	IC PIN	FUNCTION
J9-19C	Q10*	U04 - 5	OUTPUT
J9-22C	Q11*	U32 - 6	OUTPUT
J9-27B	Q10IN*	U05 - 6	OUTPUT
J9-27C	MM*	U38 - 9	OUTPUT
J9-55A	MEM*	U32 - 8	OUTPUT
J10-1A	IND	U13 - 6	OUTPUT
J10-2A	EXT1	U72 - 4	INPUT
J10-2B	LINK	U13 - 8	OUTPUT
J10-3A	EXT2	U72 - 3	INPUT
J10-4A	EXT3	U72 - 2	INPUT
J10-4B	MAR03*	U43 - 14	OUTPUT
J10-5A	PFEIN	U06 - 8	OUTPUT
J10-5B	MAR02*	U43 - 16	OUTPUT
J10-6A	HLTP*	U65 - 12	OUTPUT
J10-6B	MAR01*	U43 - 18	OUTPUT
J10-7A	EX*	U19 - 11	OUTPUT
J10-7B	MAR00*	U43 - 20	OUTPUT
J10-9A	DAT01	U37 - 17	I/O
J10-9B	ZERO	U09 - 7	OUTPUT
J10-10A	FLAG2	U71 - 8	OUTPUT
J10-10B	IR00	U25 - 11	OUTPUT
J10-11A	DAT04	U37 - 8	I/O
J10-11B	AI	U20 - 5	INPUT
J10-12A	DAT05	U37 - 7	I/O
J10-12B	IR01	U25 - 5	OUTPUT
J10-13A	DAT07	U37 - 3	I/O
J10-13B	MAR07*	U42 - 14	OUTPUT
J10-14A	I3	U19 - 8	OUTPUT

J10-15A,	ETIR*	U46 - 1	OUTPUT
J10-15B	MAR06*	U42 - 16	OUTPUT
J10-16A	I2	U56 - 4	OUTPUT
J10-16B	MAR05*	U42 - 18	OUTPUT
J10-17B	MAR04*	U42 - 20	OUTPUT
J10-18A	COND	U49 - 3	OUTPUT
J10-18B	TOR*	U20 - 8	OUTPUT
J10-19A	TEST*	U48 - 13	INPUT
J10-19B	IDR*	U20 - 11	OUTPUT
J10-20A	ELSB*	U70 - 4	OUTPUT
J10-20B	TIB*	U20 - 6	OUTPUT
J10-21A	ESPC*	U70 - 3	OUTPUT
J10-21B	DAT03	U37 - 13	I/O
J10-22A	ESTRT*	U70 - 2	OUTPUT
J10-22B	DAT02	U37 - 14	I/O
J10-23A	EBCH*	U70 - 1	OUTPUT
J10-23B	A	U13 - 1	INPUT
J10-24A	ARM	U63 - 1	INPUT
J10-25A	HALT*	U48 - 9	OUTPUT
J10-25B	READY	U09 - 17	INPUT
J10-26A	EDR	U48 - 60	OUTPUT
J10-26B	TSSR	U09 - 18	INPUT
J10-27A	IAM*	U48 - 6	OUTPUT
J10-27B	DAT00	U37 - 18	I/O
J10-29A	IR*	U48 - 3	INPUT
J10-30A	AUV	U09 - 4	OUTPUT
J10-30B	A*	U38 - 15	INPUT
J10-31A	DAT12	U36 - 8	I/O
J10-31B	DAT11	U36 - 13	I/O
J10-33A	DAT13	U36 - 7	I/O

TABLE 56 (CONT'D)

J10-33B	DAT09	U36 - 17	I/O
J10-34A	DAT10	U36 - 14	I/O
J10-34B	DAT08	U36 - 18	I/O
J10-35A	FOV	U06 - 6	OUTPUT
J10-35B	MAR11*	U40 - 14	OUTPUT
J10-36A	DAT14	U36 - 4	I/O
J10-36B	MAR10*	U40 - 16	OUTPUT
J10-37A	RPTUV*	U04 - 14	OUTPUT
J10-38A	DAT15	U36 - 3	I/O
J10-38B	MAR09*	U40 - 18	OUTPUT
J10-39A	DAT06	U37 - 4	I/O
J10-39B	MAR08*	U40 - 20	OUTPUT
J10-40A	II	U64 - 11	OUTPUT
J10-40B	IR08	U25 - 12	OUTPUT
J10-41B	SIGN	U09 - 8	OUTPUT
J10-42B	COUT	U09 - 13	OUTPUT
J10-43B	IR02	U18 - 11	OUTPUT
J10-45B	IR09	U25 - 4	OUTPUT
J10-46A	UMA9	U15 - 15	OUTPUT
J10-47A	UMA8	U15 - 19	OUTPUT
J10-48A	UMA7	U15 - 18	OUTPUT
J10-49A	UMA6	U15 - 17	OUTPUT
J10-50A	UMA5	U15 - 16	OUTPUT
J10-50B	PUS	U60 - 1	INPUT
J10-51A	UMA4	U15 - 5	OUTPUT
J10-51B	IR03	U18 - 5	OUTPUT
J10-52A	UMA3	U15 - 4	OUTPUT
J10-52B	MAR15*	U39 - 14	OUTPUT
J10-53A	UMA2	U15 - 3	OUTPUT
J10-53B	MAR14*	U39 - 16	OUTPUT
J10-54A	UMA1	U15 - 2	OUTPUT

TABLE 56 (CONT'D)

J10-54B	MAR13*	039 - 14	OUTPUT
J10-55A	UMAU	015 - 1	OUTPUT
J10-56B	MAR12*	039 - 20	OUTPUT

TABLE 56 (CONT'D)

APPENDIX E

CPU CARD DESCRIPTIONS

This Appendix contains:

- o Table of chip labels vs chip types,
- o BLISS programs for chip interconnections,
- o Table of connections between chips and card connectors, and
- o Description of each external connection for J10 CPU card.

CPU CHIP LABELS

CHIP ID		CHIP TYPE
U1	=	54-S-472
U2	=	54-LS-273
U3	=	54-S-151
U4	=	54-S-151
U5	=	54-LS-02
U6	=	54-LS-113
U7	=	54-LS-153
U8	=	54-S-472
U9	=	54-LS-273
U10	=	54-LS-175
U11	=	54-LS-253
U12	=	54-LS-86
U13	=	54-LS-113
U14	=	54-LS-153
U15	=	54-LS-472
U16	=	54-LS-273
U17	=	54-LS-175
U18	=	54-LS-153
U19	=	54-LS-08
U20	=	54-LS-00
U21	=	54-LS-169
U22	=	54-S-472
U23	=	54-LS-273
U24	=	AM-2902
U25	=	54-LS-153
U27	=	54-S-32
U28	=	54-LS-169
U29	=	AM-2901-A
U30	=	54-LS-374
U31	=	54-LS-374
U32	=	AM-2901-A
U33	=	54-LS-374
U34	=	54-LS-374
U35	=	AM-2901-A

TABLE 57

CHIP ID

CHIP TYPE

U36	=	54-LS-374
U37	=	54-LS-374
U38	=	AM-2901-A
U39	=	9407
U40	=	9407
U41	=	54-S-02
U42	=	9407
U43	=	9407
U44	=	54-LS-86
U45	=	54-S-472
U46	=	25-LS-377
U47	=	54-LS-158
U48	=	54-S-04
U49	=	54-S-00
U50	=	54-LS-352
U51	=	54-LS-352
U52	=	54-S-472
U53	=	54-LS-374
U54	=	54-LS-158
U56	=	54-LS-02
U57	=	54-LS-352
U58	=	54-LS-352
U59	=	25-S-472
U60	=	54-LS-273
U61	=	54-LS-253
U62	=	54-125
U63	=	54-S-20
U64	=	54-LS-00
U65	=	54-LS-11
U66	=	54-S-472
U67	=	54-LS-175
U68	=	54-LS-175
U69	=	54-LS-367
U70	=	54-S-288
U71	=	54-LS-113
U72	=	54-LS-151

TABLE 57 (CONT'D)

```

MODULE PIT (LANGUAGE(%BLISS36(BLISS36)
    %BLISS32(BLISS32)
    %BLISS16(BLISS16)),
    addressing_mode(external = long_relative,
        nonexternal = long_relative) ) =
BEGIN
GLOBAL ROUTINE PARTT1 : novalue =
    BEGIN    %( ADDRESS GENERATE PARTITION )%

        ! THIS PARTITION CONTAINS THE FOLLOWING COMPONENTS:
        ! THE ADDRESS PROCESSOR;
        ! THE INSTRUCTION REGISTER

    require 'GLOBAL.R32';
    require 'RTNEST.R32';

    external routine TC352,TC158,TC9407;

    OWN
        FMEA1, FMEA2, FMEA3, FMEA4, J, K;
    OWN
        DD00, DD01, DD02, DD03, DD10, DD11, DD12,
        DD13, DD20, DD21, DD22, DD23, DD30, DD31,
        DD32, DD33, CI, CO, A, GND;
    J = 0; K = 789;

    !
    ! CPU BOARD
    ! PARTITION 1

    !
    ! OUTPUT DESTINATION "SIG NAME"
    !

    ! GENERATE SELECT PULSES FOR LOW ORDER BITS
    NAND(FMEA3,U21,U22,J); JADD(J,4);

    ! U20 = 3 -> U50 = 14, U51 = 14, "FMEA3"
    ! U57 = 14, U58 = 14

```

!	OUTPUT	DESTINATION	'SIG NAME'
! SELECT LOW ORDER EIGHT BITS OF ADDRESS			
TC352(E7DIN,U20,T,Y01,IR01,IR01,DD01,GND,DD00, IR00,IR00,Y00,U24,FMEA3,F,J);			
!	U58 = 7	-> U43 = 19	"DD01"
!	U58 = 9	-> U43 = 21	"DD00"
!	VCC	-> U58 = 3, U58 = 13	
!	GND	-> U58 = 15	
TC352(E7DIN,U20,T,Y03,IR03,IR03,DD03,GND,DD02, IR02,IR02,Y02,T,FMEA3,E7DIN,J);			
!	U51 = 7	-> U43 = 15	"DD03"
!	U51 = 9	-> U43 = 17	"DD02"
!	VCC	-> U51 = 3, U51 = 13	
TC352(E7DIN,U20,T,Y05,IR05,IR03,DD11,GND,DD10, IR03,IR04,Y04,T,FMEA3,E7DIN,J);			
!	U57 = 7	-> U42 = 19	"DD11"
!	U57 = 9	-> U42 = 21	"DD10"
!	VCC	-> U57 = 3, U57 = 13	
TC352(E7DIN,U20,T,Y07,IR07,IR03,DD13,GND,DD12, IR03,IR06,Y06,T,FMEA3,E7DIN,J);			
!	U50 = 7	-> U42 = 15, U49 = 10	"DD13"
!	U50 = 9	-> U42 = 17	"DD12"
!	VCC	-> U50 = 3, U50 = 13	
! ENABLE LOGIC FOR UPPER HALF WORD ADDRESS			
NOR(FMEA1,U20N,U21N,J); NAND(A,DD13,U20N,J+4); NAND(FMEA2,U22,A,J+8); JADD(J,12);			
!	U56 = 13	-> U54 = 1, U47 = 1	"FMEA1"
!	U49 = 8	-> U49 = 13	
!	U49 = 11	-> U54 = 15, U47 = 15	"FMEA2"

TABLE 58 (CONT'D)

!	OUTPUT	DESTINATION	'SIG NAME'
---	--------	-------------	------------

! UPPER HALF WORD ADDRESS SELECT

TC158(FMEA1,T,Y15,DD33,T,Y14,DD32,GND,DD31,
Y13,T,DD30,Y12,T,FMEA2,J);

!	U54 = 4	-> U39 = 15	"DD33"
!	U54 = 7	-> U39 = 17	"DD32"
!	U54 = 9	-> U39 = 19	"DD31"
!	U54 = 12	-> U39 = 21	"DD30"
!	VCC	-> U54 = 2, U54 = 5	
!		U54 = 11, U54 = 14	

TC158(FMEA1,T,Y11,DD23,T,Y10,DD22,GND,DD21,
Y09,T,DD20,Y08,T,FMEA2,J);

!	U47 = 4	-> U40 = 15	"DD23"
!	U47 = 7	-> U40 = 17	"DD22"
!	U47 = 9	-> U40 = 19	"DD21"
!	U47 = 12	-> U40 = 21	"DD20"
!	VCC	-> U47 = 2, U47 = 5,	
!		U47 = 11, U47 = 14	

! GENERATE MICRO SELECT FOR 9407

ANDR(I3,U23,U21,J); ORR(A,U21N,U24,J +4);
NAND(I1,A,U23N,J+8); JADD(J,12);

!	U19 = 8	-> U39 = 5, U40 = 5,	"I3"
!		U42 = 5, U43 = 5,	
!		P10 = 14A	
!	U27 = 11	-> U64 = 12	
!	U64 = 11	-> U39 = 3, U40 = 3,	"I1"
!		U42 = 3, U43 = 3,	
!		P10 = 40A	

TABLE 58 (CONT'D)

!

OUTPUT

DESTINATION

'SIG NAME'

! EXECUTE 9407 MEMORY ADDRESS GENERATE

TC9407(EXN,F,I1,I2,I3,EDR,ANCLK,D00,D01,D02,D03,
 ,K,CO,MAR03,DD03,MAR02,DD02,MAR01,DD01,
 MAR00,DD00,F,RP02,J);

!	U41 - 4	->	U39 - 7, U17 - 9, U29 - 15,	"A*"
!			U40 - 7, U16 - 11, U32 - 15,	
!			U42 - 7, U10 - 9, U35 - 15,	
!			U43 - 7, U23 - 11, U38 - 15,	
!			U2 - 11, U53 - 11, U60 - 11,	
!			U9 - 11, U67 - 9, U68 - 9,	
!			U28 - 2, U21 - 2, U46 - 11,	
!			P09 - 30B	
!	GND	->	U41 - 6	

!	P10 - 8B	->	U39 - 22, U40 - 22, U42 - 22,	"QMAR"
!			U43 - 22	

!	U43 - 13	->	U42 - 23	
!	U43 - 14	->	P10 - 4B	"MAR03*"
!	U43 - 16	->	P10 - 5B	"MAR02*"
!	U43 - 18	->	P10 - 6B	"MAR01*"
!	U43 - 20	->	P10 - 7B	"MAR00*"

!	VCC	->	U43 - 23	
---	-----	----	----------	--

TC9407(EXN,F,I1,I2,I3,EDR,ANCLK,D04,D05,D06,D07,
 ,K+33,CI,MAR07,DD13,MAR06,DD12,MAR05,DD11,
 MAR04,DD10,F,CO,J);

!	U42 - 13	->	U40 - 23	
!	U42 - 14	->	P10 - 13B	"MAR07*"
!	U42 - 16	->	P10 - 15B	"MAR06*"
!	U42 - 18	->	P10 - 16B	"MAR05*"
!	U42 - 20	->	P10 - 17B	"MAR04*"

TC9407(EXN,F,I1,I2,I3,EDR,ANCLK,D08,D09,D10,D11,
 ,K+66,CO,MAR11,DD23,MAR10,DD22,MAR09,DD21,
 MAR08,DD20,F,CI,J);

!	U40 - 13	->	U39 - 23	
!	U40 - 14	->	P10 - 35B	"MAR11*"
!	U40 - 16	->	P10 - 36B	"MAR10*"
!	U40 - 18	->	P10 - 38B	"MAR09*"
!	U40 - 20	->	P10 - 39B	"MAR08*"

TABLE 58 (CONT'D)

!	OUTPUT	DESTINATION	'SIG NAME'
---	--------	-------------	------------

```

TC9407(EXN,F,I1,I2,I3,EDR,ANCLK,D12,D13,D14,D15,
      ,K+99,A,MAR15,DD33,MAR14,DD32,MAR13,
      DD31,MAR12,DD30,F,CO,J);

```

!	U39 - 14	-> P10 - 52B	"MAR15*"
!	U39 - 16	-> P10 - 53B	"MAR14*"
!	U39 - 18	-> P10 - 54B	"MAR13*"
!	U39 - 20	-> P10 - 56B	"MAR12*"

```

      END;    %( ROUTINE PARTN1 )%
END
ELUDOM    %( P10 )%

```

TABLE 58 (CONT'D)

ANDR(EXN,E7DIN,U20N); NOR(I2,U23N,U24N);

!	U19 - 11	->	U39 = 1, U40 = 1, U42 = 1,	"EX*"
!			U43 = 1, U65 = 13, P10 = 7A,	
!	U56 - 4	->	U39 = 4, U40 = 4, U42 = 4,	"I2"
!			I43 = 4, P10 = 16A,	

EDRN = AND3(U24,U20N,E7DIN); INV(EDR,EDRN);

!	U65 = 8	->	U36 = 1, U37 = 1, U48 = 5	"EDR*"
!	U48 = 6	->	U39 = 6, U40 = 6, U42 = 6,	"EDR"
!			U43 = 6, P10 = 26A	

! PROGRAM COUNTER FEEDBACK PORTION OF 9407S

KK = 789; ! SET STORAGE POINTER TO PARTITION 1

IC257(I2,ST[.KK],ST[.KK+4],D00,ST[.KK+1],ST[.KK+5],D01,.KK+32,
D02,ST[.KK+6],ST[.KK+2],D03,ST[.KK+7],ST[.KK+3],EDR);

TC257(I2,ST[.KK+33],ST[.KK+37],D04,ST[.KK+34],ST[.KK+38],D05,.KK+65,
D06,ST[.KK+39],ST[.KK+35],D07,ST[.KK+40],ST[.KK+36],EDR);

TC257(I2,ST[.KK+66],ST[.KK+70],D08,ST[.KK+67],ST[.KK+71],D09,.KK+98,
D10,ST[.KK+72],ST[.KK+68],D11,ST[.KK+73],ST[.KK+69],EDR);

TC257(I2,ST[.KK+99],ST[.KK+103],D12,
ST[.KK+100],ST[.KK+104],D13,.KK+131,
D14,ST[.KK+105],ST[.KK+101],D15,
ST[.KK+106],ST[.KK+102],EDR);

! MEMORY DATA REGISTER

TC374(EDRN,TT07,DAT07,DAT06,TT06,TT05,DAT05,DAT04,TT04,.K+32,
IDRN,TT03,DAT03,DAT02,TT02,TT01,DAT01,DAT00,TT00);

!	U37 = 2	->	U32 = 22, U42 = 11	"D07"
!	U37 = 5	->	U32 = 23, U42 = 10	"D06"
!	U37 = 6	->	U32 = 24, U42 = 09	"D05"
!	U37 = 9	->	U32 = 25, U42 = 08	"D04"
!	U37 = 12	->	U38 = 22, U43 = 11	"D03"
!	U37 = 15	->	U38 = 23, U43 = 10	"D02"
!	U37 = 16	->	U38 = 24, U43 = 09	"D01"
!	U37 = 19	->	U38 = 25, U43 = 08	"D00"

TC374(EDRN,TT15,DAT15,DAT14,TT14,TT13,DAT13,DAT12,TT12,.K+48,

1DRN,TT11,DAT11,DAT10,TT10,TT09,DAT09,DAT08,TT08);

!	U36 - 2	->	U35 - 22, U39 - 11	"D15"
!	U36 - 5	->	U35 - 23, U39 - 10	"D14"
!	U36 - 6	->	U35 - 24, U39 - 9	"D13"
!	U36 - 9	->	U35 - 25, U39 - 8	"D12"
!	U36 - 12	->	U29 - 22, U40 - 11	"D12"
!	U36 - 15	->	U29 - 23, U40 - 10	"D10"
!	U36 - 16	->	U29 - 24, U40 - 9	"D09"
!	U36 - 19	->	U29 - 25, U40 - 8	"D08"

D00=,D00 AND .TT00; D01=,D01 AND .TT01; D02=,D02 AND .TT02;
D03=,D03 AND .TT03; D04=,D04 AND .TT04; D05=,D05 AND .TT05;
D06=,D06 AND .TT06; D07=,D07 AND .TT07; D08=,D08 AND .TT08;
D09=,D09 AND .TT09; D10=,D10 AND .TT10; D11=,D11 AND .TT11;
D12=,D12 AND .TT12; D13=,D13 AND .TT13; D14=,D14 AND .TT14;
D15=,D15 AND .TT15;
IF ,JPART EQL 4 THEN ALU() ELSE ALUT();

TABLE 59 (CONT'D)

! TIMING SIGNALS FOR MEMORY FETCH AND STORE

!	P09 - 21B ->	U32 = 9, U38 = 1, U13 = 9,	"U27*"
!		U44 = 4, U38 = 13	
!	P09 - 23C ->	U32 = 1, U32 = 5	"U27"
!	P09 - 24C ->	U32 = 2, U13 = 13	"U28*"
!	P09 - 21C ->	U32 = 4, U38 = 2, U32 = 10,	"U28"
!		U44 = 1, U38 = 10	
!	P09 - 25C ->	U13 = 2	"U30*"
!	P09 - 4C ->	U13 = 11	"U30"
!	P09 - 18C ->	U13 = 1, U13 = 10, U44 = 2	"U29"
!	P09 - 15C ->	U44 = 5	"IND*"

NAND(MEMN,U28,U27N); NAND3(A,U28N,U29,U30N);

!	U32 = 8 ->	P09 = 55A	"MEM*"
!	U13 = 12 ->	U04 = 3	

NAND(QION,U27,U28N); NAND3(QIOINN,U27N,U29,U30);

!	U32 = 3 ->	U04 = 5, P09 = 19C	"QIO*"
!	U13 = 8 ->	U05 = 5	
!	U32 = 6 ->	U05 = 4, P09 = 22C	"QII*"
!	U05 = 6 ->	P09 = 27B	"QIOIN*"

NAND4(MMN,U27N,U28,U29,INDN);

!	U44 = 6 ->	U38 = 9, U04 = 4, P09 = 27C	"MM*"
---	------------	-----------------------------	-------

EDUTN = AND3(QION,A,MMN); INV(MM,MMN);

!	U04 = 6 ->	P09 = 30C	"EOUT*"
---	------------	-----------	---------

NAND3(QMIN,U27N,U28,MMN);

!	U38 = 8 ->	P09 = 25A	"QMI*"
---	------------	-----------	--------

NOR(A, U28N,U30N); NAND4(B,U27N,U30N,QMIN,T);

!	U56 = 1 ->	U20 = 4	
!	P10 = 17A ->	U63 = 12	
!	U63 = 8 ->	U20 = 13	
!	VCC ->	U63 = 13	

TABLE 59 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
!	THESE NAND GATES ARE SIMULATED WITHOUT THE INVERTER ON		
!	THE OUTPUT SO THAT THE CLOCK SIGNAL IS CORRECT FOR THE		
!	SINGLE PASS SIMULATION OF A D FLIP-FLOP10		

ANDR(NTIBN,AICLK,A); ANDR(NTORN,AICLK,U19);

!	P10 - 11B ->	U20 - 5, U20 - 12,	"AI"
!		U41 - 5, U65 - 4	
!	U20 - 6 ->	U33 - 11, U34 - 11, P10 - 20B	"TIB*"
!	U20 - 8 ->	U30 - 11, U31 - 11, P10 - 18B	"TOR*"

ANDR(NTDRN,AICLK,B);

!	U20 - 11 ->	U36 - 11, U37 - 11, P10 - 19B	"TDR*"
---	-------------	-------------------------------	--------

NAND4(MIN,U27N,U28,BN,POS);

! MEMORY BUFFER REGISTER

TC374(U40,TT07,DAT07,DAT06,TT06,TT05,DAT05,DAT04,TT04,.K,
TIBN,TT03,DAT03,DAT02,TT02,TT01,DAT01,DAT00,TT00);

TC374(U40,TT15,DAT15,DAT14,TT14,TT13,DAT13,DAT12,TT12,.K+16,
TIBN,TT11,DAT11,DAT10,TT10,TT09,DAT09,DAT08,TT08);

YN00=.YN00 AND .TT00; YN01=.YN01 AND .TT01; YN02=.YN02 AND .TT02;
YN03=.YN03 AND .TT03; YN04=.YN04 AND .TT04; YN05=.YN05 AND .TT05;
YN06=.YN06 AND .TT06; YN07=.YN07 AND .TT07; YN08=.YN08 AND .TT08;
YN09=.YN09 AND .TT09; YN10=.YN10 AND .TT10; YN11=.YN11 AND .TT11;
YN12=.YN12 AND .TT12; YN13=.YN13 AND .TT13; YN14=.YN14 AND .TT14;
YN15=.YN15 AND .TT15;

TABLE 59 (CONT'D)

!

OUTPUT

DESTINATION

"SIG NAME"

! OUTPUT REGISTER

TC374(EOUTN,DAT07,Y07,Y06,DAT06,DAT05,Y05,Y04,DAT04,,K+64,
IORN,DAT03,Y03,Y02,DAT02,DAT01,Y01,Y00,DAT00);

!	P10 - 29B ->	U30 - 1, U31 - 1	
!	U31 - 2 ->	U34 - 3, U37 - 3, U71 - 12,	"DAT07"
!		P10 - 13A	
!	U31 - 5 ->	U34 - 4, U37 - 4, U71 - 11,	"DAT06"
!		P10 - 39A	
!	U31 - 6 ->	U34 - 7, U37 - 7, U71 - 2,	"DAT05"
!		P10 - 12A	
!	U31 - 9 ->	U34 - 8, U37 - 8, U71 - 3,	"DAT04"
!		P10 - 11A	
!	U31 - 12 ->	U34 - 13, U37 - 13, U19 - 1,	"DAT03"
!		P10 - 21B	
!	U31 - 15 ->	U34 - 14, U37 - 14, U27 - 9,	"DAT02"
!		P10 - 22B	
!	U31 - 16 ->	U34 - 17, U37 - 17, U07 - 5,	"DAT01"
!		P10 - 9A	
!	U31 - 19 ->	U34 - 18, U37 - 18, U07 - 11,	"DAT00"
!		P10 - 27B	

TABLE 59 (CONT'D)

!

OUTPUT

DESTINATION

"SIG NAME"

TC374(EOUTN,DAT15,Y15,Y14,DAT14,DAT13,Y13,Y12,DAT12,,K+80,
TURN,DAT11,Y11,Y10,DAT10,DAT09,Y09,Y08,DAT08);

!	U30 - 2 ->	U33 - 3, U36 - 3, P10 - 38A	"DAT15"
!	U30 - 5 ->	U33 - 4, U36 - 4, P10 - 36A	"DAT14"
!	U30 - 6 ->	U33 - 7, U36 - 7, P10 - 33A	"DAT13"
!	U30 - 9 ->	U33 - 8, U36 - 8, P10 - 31A	"DAT12"
!	U30 - 12 ->	U33 - 13, U36 - 13, P10 - 31B	"DAT11"
!	U30 - 15 ->	U33 - 14, U36 - 14, P10 - 34A	"DAT10"
!	U30 - 16 ->	U33 - 17, U36 - 17, P10 - 33B	"DAT09"
!	U30 - 19 ->	U33 - 18, U36 - 18, P10 - 34B	"DAT08"

! MEMORY BUFFERS AND MEMORY CALLS

TC2450(MMN,M00,M01,M02,M03,M04,M05,M06,M07,GND,DAT07,
DAT06,DAT05,DAT04,DAT03,DAT02,DAT01,DAT00,MMN);

TC2450(MMN,M08,M09,M10,M11,M12,M13,M14,M15,GND,DAT15,
DAT14,DAT13,DAT12,DAT11,DAT10,DAT09,DAT08,MMN);

!

TABLE 59 (CONT'D)


```
RWMEM(MAR00,MAR01,MAR02,MAR03,MAR04,MAR05,MAR06,MAR07,
      MAR08,MAR09,MAR10,MAR11,MAR12,MAR13,MAR14,MAR15,MMN,
      MIN,M00,M01,M02,M03,M04,M05,M06,M07,M08,M09,M10,M11,
      M12,M13,M14,M15);
```

!

```
TC245I(MMN,M00,M01,M02,M03,M04,M05,M06,M07,GND,TT07,
      TT06,TT05,TT04,TT03,TT02,TT01,TT00,MMN);
```

!

```
TC245I(MMN,M08,M09,M10,M11,M12,M13,M14,M15,GND,TT15,
      TT14,TT13,TT12,TT11,TT10,TT09,TT08,MMN);
```

!

```
DAT00=.DAT00 AND .QIOINN AND .TT00;
DAT01=.DAT01 AND .QIOINN AND .TT01;
DAT02=.DAT02 AND .QIOINN AND .TT02;
DAT03=.DAT03 AND .QIOINN AND .TT03;
DAT04=.DAT04 AND .QIOINN AND .TT04;
DAT05=.DAT05 AND .QIOINN AND .TT05;
DAT06=.DAT06 AND .QIOINN AND .TT06;
DAT07=.DAT07 AND .QIOINN AND .TT07;
DAT08=.DAT08 AND .QIOINN AND .TT08;
DAT09=.DAT09 AND .QIOINN AND .TT09;
DAT10=.DAT10 AND .QIOINN AND .TT10;
DAT11=.DAT11 AND .QIOINN AND .TT11;
DAT12=.DAT12 AND .QIOINN AND .TT12;
DAT13=.DAT13 AND .QIOINN AND .TT13;
DAT14=.DAT14 AND .QIOINN AND .TT14;
DAT15=.DAT15 AND .QIOINN AND .TT15;
```

!

IND AND LINK FLIP-FLOP

```
TC153(U33,U34,YN15,RPU1,F,IND,YA,GND,YB,SRAM,F,
      RPU1,QBIT,U35,U33N);
```

!	U38	=	9	->	U14	=	10,	U35	=	16		"SRAM00"
!	U14	=	7	->	U05	=	8,	U13	=	2		
!	U14	=	9	->	U05	=	11,	U13	=	12		
!	VCC			->	U14	=	4,	U14	=	12		
!	GND			->	U14	=	5,	U14	=	11		

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

NOR(A,YA,U33); NOR(B,YB,U33N);

!	U5 - 10 ->	U13 - 3	
!	U5 - 13 ->	U13 - 11	

TC113(ACLK,YA,A,RPU1,NINDN,NIND,GND,NLINK,NLINKN,RPU1,
B,YB,ACLK);

!	P10 - 23B ->	U13 - 1, U13 - 13, U6 - 13,	"A"
!		U06 - 1	
!	U13 - 0 ->	U04 - 3, U11 - 4, U12 - 4,	
!		U14 - 6, U44 - 5, P10 - 1A	
!	U13 - 5 ->	P10 - 3B	"IND*"
!	U13 - 8 ->	U04 - 2, P10 - 2B	"LINK"
!	VCC ->	U13 - 4, U13 - 10	

! OVERFLOW FLAG COMPUTATION

XORR(A,YN14,YN15); INV(B,AOV); NOR(C,U32,U31);

!	U12 - 3 ->	U7 - 3	
!	U35 - 34 ->	U48 - 1, U7 - 6, U44 - 2,	"AOV"
!		U9 - 4, P10 - 30A	
!	U48 - 2 ->	U7 - 10	
!	U05 - 1 ->	U7 - 15	

TC153(U31N,U32,A,U31,DAT01,AOV,YA,GND,YB,B,DAT00,U31N,
F,U33,C);

!	U07 - 7 ->	U06 - 2	
!	U07 - 9 ->	U06 - 3	
!	GND ->	U07 - 13	

! INTERRUPT ENABLE FLIP-FLOP

NOR(A,U32,U33N); ORR(B,DAT02,U31N); ANDR(C,DAT03,A);

!	U05 - 4 ->	U19 - 2, U19 - 4, U64 - 1,	
!		U65 - 3	
!	U27 - 8 ->	U19 - 5	
!	U19 - 3 ->	U06 - 12	

ANDR(D,A,B); NAND(E,U31N,A);

!	U19 - 6 ->	U06 - 11	
!	U64 - 3 ->	U62 - 1	

TABLE 59 (CONT'D)

```

TC113(ACLK,YA,YB,RPU1,NFOVN,NFOV,GND,NPFEIN,NNPFEIN,RPU1,
D,C,ACLK);

```

!	U06 -	5 ->	U04 -	1		"FOV*"
!	U06 -	6 ->	U72 -	15, P10 -	35A	"FOV"
!	U06 -	8 ->	U63 -	5, U72 -	1, P10 -	"PFEI*"
!	U06 -	9 ->	U41 -	12		"NPFEI*"
!	VCC	->	U06 -	4, U06 -	10	

! FLAG1 AND FLAG2 FLIP-FLOP

```

B = AND3(A,AICLK,U31); C = NOT .E; ANDR(IAMN,T,C);

```

!	U65 -	6 ->	U71 -	1, U71 -		
!	U62 -	3 ->	P10 -	27A		"IAM*"
!	GND	->	U62 -	2		

```

IC113(B,DAT05,DAT04,RPU1,NN1FLAG,NFLAG1,GND,NFLAG2,NN2FLAG,
RPU1,DAT06,DAT07,B);

```

!	U71 -	6 ->	U72 -	13, P10 -		"FLAG1"
!	U71 -	8 ->	U72 -	12, P10 -	10A	"FLAG2"
!	VCC	->	U71 -	4, U71 -	10	

```

END;     !   ROUTINE PARTN2
END
ELUDOM   ! P2

```

```

MODULE P3T (LANGUAGE(%BLISS36(BLISS36)
    %BLISS32(BLISS32)
    %BLISS16(BLISS16)),
    addressing_mode (external = long_relative,
        nonexternal = long_relative) ) =
BEGIN
GLOBAL ROUTINE PARTT3 : novalue =
    BEGIN    %(  FETCH PARTITION  )%

    ! THIS PARTITION PERFORMS THE FETCHING PORTION OF AN
    ! INSTRUCTION CYCLE OF THE BDX-930 COMPUTER.
    ! IT CONTAINS THE FOLLOWING PARTS OF
    ! THE COMPUTER:
    !     ADDRESS CONTROL LOGIC
    !     INSTRUCTION REGISTER AND REPEAT COUNTER
    !     MICROPROGRAM COUNTER

    require "GLOBAL.R32";
    require "RTINEST.R32";
    EXTERNAL RU : vector[57];
    external routine TC151,TC273,TC253,TC175,TC169,TC377,TC374,PENABLE,
        PFIRST,PMIM;

    LOCAL  A,  B,  C,  J,  K,  GND,  X,  NC;
    LOCAL  CON,  CGND,  EBCHN,  ESTRTN,  ESPCN,  ELSBN,  EMSBN,
        ETIRN,  IRS,  FEIN,  SA,  SB,  HALTMN;
OWN      UMA2DE,  UMA3DE,  UMA4DE,  UMA5DE,  UMA6DE,  UMA7DE,  UMA8DE,
        UMA9DE;
OWN      UMA0DI,  UMA1DI,  UMA2DI,  UMA3DI,  UMA4DI,  UMA5DI,  UMA6DI,
        UMA7DI;

OWN      UMA0,  UMA1,  UMA2,  UMA3,  UMA4,  UMA5,  UMA6,  UMA7,  UMA8,
        UMA9;
OWN      UMA0DN,  UMA1DN,  UMA2DN,  UMA3DN,  UMA4DN,  UMA5DN,  UMA6DN,
        UMA7DN,  UMA8DN,  UMA9DN;

    J = 0; NC = -1; K = 616;

```

TABLE 60

```

!
!           CPU BOARD
!           PARTITION 3

```

```

!           OUTPUT           DESTINATION           "SIG NAME"

```

```

!
!           REPEAT COUNTER
NAND(A, U25, U26);

```

```

!           U64 - 5 ->      U28 - 9

```

```

TC169(U26,ANCLK,Y00,Y01,Y02,Y03,U25,.K+164,A,F,
      X,X,X,X,RPTOVN);

```

```

!           GND           ->      U28 - 10
!           U28 - 15 ->      U04 - 14, U61 - 4,           "RPTOV*"
!                               P10 - 37A

```

```

!
!           SELECT CONDITION FOR BRANCH INSTRUCTIONS
TC151(PFEIN,EXT3,EXT2,EXT1,X,A,F,GND,SPB02,SPB01,
      SPB00,FLAG2,FLAG1,IRS,FOV);

```

```

!           U72 - 6 ->      U12 - 10
!           GND           ->      U72 - 7
!           P10 - 2A ->      U72 - 4           "EXT1"
!           P10 - 3A ->      U72 - 3           "EXT2"
!           P10 - 4A ->      U72 - 2           "EXT3"

```

TABLE 60 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
	XORR(CUN,SPB03,A);		
!	U12 = 8 ->	U09 = 14	"CON"
	HLILP = AND3(U23,U24N,EXN);		
!	U65 = 12 ->	P10 = 6A	"HLILP"
!	MICROCODE BRANCH CONDITION GENERATORS		
	TC151(FUVN,LINK,IND,RPU,X,A,U36,GND,U37, U38,U39,NC,SATN,RPTOVN,SQ00);		
!	P10 = 24B ->	U04 = 13	"SAT*"
!	U04 = 6 ->	U49 = 1	
!	VCC ->	U04 = 4	
	TC151(04,03,02,NC,X,B,U36N,GND, U37,U38,U39,08,07,06,05);		
!	U03 = 6 ->	U49 = 2	
	NAND(COND,A,B);		
!	U49 = 3 ->	U70 = 11, U12 = 13, P10 = 18A	"COND"
!	ENABLE PROM = MICROCODE BRANCH TABLE		
	NAND(HALTMN,HALTS,U42); NAND4(B,ARM,FEIN,IRS,PFEIN);		
!	U64 = 6 ->	U49 = 4, U61 = 13	"HALTM*"
	NAND(C,HALTMN,B); INV(B,TESTN);		
!	P10 = 24A ->	U63 = 1	"ARM"
!	U63 = 6 ->	U49 = 5	
!	U49 = 6 ->	U70 = 10	
!	P10 = 19A ->	U48 = 13	"TEST*"
!	U48 = 12 ->	U70 = 15	

TABLE 60 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

PENARLE(EBCHN,ESTRTN,ESPCN,ELSDN,EMSDN,SB,SA,ETIRN,
C,COND,U42,U41,U40,B);

!	U70 - 1 ->	U53 - 1, P10 - 23A	"ERCH"
!	U70 - 2 ->	U45 - 15, P10 - 22A	"ESIRT"
!	U70 - 3 ->	U69 - 1, U69 - 15, P10 - 21A	"ESPC"
!	U70 - 4 ->	U61 - 1, U61 - 15, P10 - 20A	"ELSD"
!	U70 - 5 ->	U62 - 4, U62 - 10	"EMSD"
!	U70 - 6 ->	U61 - 2	"SB"
!	U70 - 7 ->	U61 - 14, U41 - 2	"SA"
!	U70 - 9 ->	U46 - 1, U21 - 9, P10 - 15A	"ETIR"

! MICROCODE BRANCH ADDRESS GENERATOR

XORR(A,COND,U52N);

! U12 - 11 -> U61 - 11, U61 - 12

TC253(ELSDN,SB,U41,RPIOVN,U51,U51,UMA1DN,GND,UMA0DN,U52,
A,A,HALTMN,SA,ELSDN);

!	U61 - 7 ->	U45 - 7, U15 - 2, U52 - 2,	"UMA1"
!		U08 - 2, U59 - 2, U22 - 2,	
!		U66 - 2, U01 - 2, P10 - 54A	
!	U61 - 9 ->	U45 - 6, U15 - 1, U52 - 1,	"UMA0"
!		U08 - 1, U59 - 1, U22 - 1,	
!		U66 - 1, U01 - 1, P10 - 55A	

! GND -> U69 - 2, U69 - 4, U69 - 6, U69 - 10,
! U69 - 12, U69 - 14

BUF(UMA2DN,ESPCN); BUF(UMA3DN,ESPCN);

!	U69 - 9 ->	U45 - 8, U15 - 3, U52 - 3,	"UMA2"
!		U08 - 3, U59 - 3, U22 - 3,	
!		U66 - 3, U01 - 3, U53 - 19,	
!		P10 - 53A	
!	U69 - 7 ->	U45 - 9, U15 - 4, U52 - 4,	"UMA3"
!		U08 - 4, U59 - 4, U22 - 4,	
!		U66 - 4, U01 - 4, U53 - 16,	
!		P10 - 52A	

TABLE 60 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

BUF(UMA4DN,ESPCN); BUF(UMA5DN,ESPCN);

!	U69 - 5 ->	U45 - 11, U15 - 5, U52 - 5,	"UMA4"
!		U08 - 5, U59 - 5, U22 - 5,	
!		U66 - 5, U01 - 5, U53 - 15,	
!		P10 - 51A	
!	U69 - 3 ->	U45 - 12, U15 - 16, U52 - 16,	"UMA5"
!		U08 - 16, U59 - 16, U22 - 16,	
!		U12 - 16, U66 - 16, U53 - 12,	
!		P10 - 50A	

BUF(UMA6DN,ESPCN); BUF(UMA7DN,ESPCN);

!	U69 - 11 ->	U45 - 13, U15 - 17, U52 - 17,	"UMA6"
!		U08 - 17, U59 - 17, U22 - 17,	
!		U66 - 17, U01 - 17, U53 - 9,	
!		P10 - 49A	
!	U69 - 13 ->	U45 - 14, U15 - 18, U52 - 18,	"UMA7"
!		U08 - 18, U59 - 18, U22 - 18,	
!		U66 - 18, U01 - 18, U53 - 6,	
!		P10 - 48A	
!	GND ->	U62 - 5, U62 - 9	

BUF(UMA8DN,EMSBN); BUF(UMA9DN,EMSBN);

!	U62 - 8 ->	U15 - 19, U52 - 19, U53 - 5,	"UMA8"
!		U08 - 19, U22 - 19, U59 - 19,	
!		U66 - 19, U01 - 19, P10 - 47A	
!	U62 - 6 ->	U15 - 15, U52 - 15, U53 - 2,	"UMA9"
!		U08 - 15, U22 - 15, U59 - 15,	
!		U66 - 15, U01 - 15, P10 - 46A	

TC374(EBCHN,UMA9DE,RU[14],RU[13],UMA8DE,UMA7DE,RU[12],
RU[11],UMA6DE,.K+16,ANCLK,UMA5DE,RU[10],RU[9],UMA4DE,
UMA3DE,RU[8],RU[7],UMA2DE);

PFIRST(Y08,Y09,Y10,Y11,Y12,UMA0DI,UMA1DI,UMA2DI,UMA3DI,
UMA4DI,UMA5DI,UMA6DI,UMA7DI,ESTRTN,Y13,Y14,Y15,U41N);

TABLE 60 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

! SIMULATE TRI STATE BUS FOR MICRO MEMORY ADDRESS

```

UMA0 = .UMA0DN AND .UMA0DI;
UMA1 = .UMA1DN AND .UMA1DI;
UMA2 = .UMA2DN AND .UMA2DI AND .UMA2DE;
UMA3 = .UMA3DN AND .UMA3DI AND .UMA3DE;
UMA4 = .UMA4DN AND .UMA4DI AND .UMA4DE;
UMA5 = .UMA5DN AND .UMA5DI AND .UMA5DE;
UMA6 = .UMA6DN AND .UMA6DI AND .UMA6DE;
UMA7 = .UMA7DN AND .UMA7DI AND .UMA7DE;
UMA8 = .UMA8DN AND .UMA8DE;
UMA9 = .UMA9DN AND .UMA9DE;

```

! THIS IS FUNCTIONAL - REPRESENTING PROMS:
! U52,U59,U66,U1,U22,U8,AND, U15 - OUTPUTS ARE IN RU15

```

!      U52 - 6 -> U68 - 13
!      U52 - 7 -> U60 - 18
!      U52 - 8 -> U53 - 7
!      U52 - 9 -> U53 - 8
!      U52 - 11 -> U53 - 13
!      U52 - 12 -> U53 - 14
!      U52 - 13 -> U53 - 17
!      U52 - 14 -> U53 - 18

```

```

!      U59 - 6 -> U60 - 13
!      U59 - 7 -> U60 - 17
!      U59 - 8 -> U60 - 3
!      U59 - 9 -> U60 - 4
!      U59 - 11 -> U60 - 7
!      U59 - 12 -> U60 - 8
!      U59 - 13 -> U53 - 3
!      U59 - 14 -> U53 - 4

```

```

!      U66 - 6 -> U68 - 12
!      U66 - 7 -> U60 - 14
!      U66 - 8 -> U67 - 4
!      U66 - 9 -> U67 - 5
!      U66 - 11 -> U67 - 12
!      U66 - 12 -> U67 - 13
!      U66 - 13 -> U68 - 4
!      U66 - 14 -> U68 - 5

```

```

!      U01 - 6 -> U02 - 3
!      U01 - 7 -> U02 - 4
!      U01 - 8 -> U02 - 7
!      U01 - 9 -> U02 - 8
!      U01 - 11 -> U02 - 13
!      U01 - 12 -> U02 - 14
!      U01 - 13 -> U02 - 17
!      U01 - 14 -> U02 - 18

```

```

!      U22 - 6 -> U23 - 3

```

TABLE 60 (CONT'D)

```

!      U22 = 7 -> U23 = 4
!      U22 = 8 -> U23 = 7
!      U22 = 9 -> U23 = 8
!      U22 = 11 -> U23 = 13
!      U22 = 12 -> U23 = 14
!      U22 = 13 -> U23 = 17
!      U22 = 14 -> U23 = 18

```

```

!      U08 = 6 -> U16 = 13
!      U08 = 7 -> U16 = 14
!      U08 = 8 -> U16 = 17
!      U08 = 9 -> U16 = 18
!      U08 = 11 -> U10 = 4
!      U08 = 12 -> U10 = 5
!      U08 = 13 -> U10 = 12
!      U08 = 14 -> U10 = 13

```

```

!      U15 = 6 -> U17 = 4
!      U15 = 7 -> U17 = 5
!      U15 = 8 -> U17 = 12
!      U15 = 9 -> U17 = 13
!      U15 = 11 -> U16 = 3
!      U15 = 12 -> U16 = 4
!      U15 = 13 -> U16 = 7
!      U15 = 14 -> U16 = 8

```

PMIM(UMA0,UMA1,UMA2,UMA3,UMA4,UMA9,UMA5,UMA6,UMA7,UMA8);

! INDEX REGISTER COUNTER

TC169(RPU2,ANCLK,Y04,Y05,Y06,Y07,U53,.K+156,ETIRN,U53,IR07,
IR06,IR05,IR04,X);

```

!      U21 = 11 ->      U50 = 5, U18 = 6           "IR07"
!      U21 = 12 ->      U50 = 11, U18 = 10        "IR06"
!      U21 = 13 ->      U57 = 5, U25 = 6          "IR05"
!      U21 = 14 ->      U57 = 11, U25 = 10        "IR04"
!      VCC      ->      U21 = 1

```

! INSTRUCTION REGISTER

NOR(A,FEIN,SA); NOR(B,A,PFEINN);
INV(A,HALTN);

```

!      P10 = 25A ->      U48 = 9           "HALT*"
!      U41 = 1 ->      U41 = 11
!      U41 = 13 ->      U46 = 17
!      U48 = 6 ->      U46 = 18

```

TABLE 60 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

TC377(ETIRN,IR09,Y09,Y08,IR08,IR03,Y03,Y02,IR02,
.K+124,ANCLK,IR01,Y01,Y00,IR00,FEIN,B,A,HALTS);

!	U46 - 2 ->	U25 - 4, P10 - 45B	"IR09"
!	U46 - 5 ->	U25 - 12, P10 - 40B	"IR08"
!	U46 - 6 ->	U18 - 5, U29 - 1,	"IR03"
!		U32 - 1, U35 - 1,	
!		U38 - 1, U51 - 5, U51 - 6,	
!		U57 - 6, U57 - 10, U50 - 6,	
!		U50 - 10, P10 - 51B	
!	U46 - 9 ->	U18 - 11, U51 - 10,	"IR02"
!		U51 - 11, U29 - 2,	
!		U32 - 2, U35 - 2, U38 - 2,	
!		P10 - 43B	
!	U46 - 12 ->	U25 - 5, U58 - 5,	"IR01"
!		U58 - 6, U29 - 3,	
!		U32 - 3, U35 - 3, U38 - 3,	
!		P10 - 12B	
!	U46 - 15 ->	U25 - 11, U58 - 10,	"IR00"
!		U58 - 11, U27 - 2,	
!		P10 - 10B	
!	U46 - 16 ->	U63 - 2, U41 - 3	"FEIN"
!	U46 - 19 ->	U64 - 4	"HALTS"

! EXTERNAL SIGNAL SYNCHRONIZER LATCH

INV(A,IRN):

!	P10 - 29A ->	U48 - 3	"IR*"
!	U48 - 4 ->	U09 - 3	

TABLE 60 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

```
TC273(RPU,IRS,A,AUV,02,03,ZERO,SIGNV,
      04,,K,ANCLK,05,COUT,CON,06,07,
      READY,TSSR,08);
```

!	U35 - 11 ->	U09 - 7, U29 - 11, U32 - 11	"ZERO"
!		U38 - 11, P10 - 9B	
!	U35 - 31 ->	U09 - 8, U11 - 11,	"SIGN"
!		U44 - 1, U12 - 5, P10 - 41B	
!	U35 - 33 ->	U09 - 13, P10 - 42B	"COUT"
!	P10 - 25B ->	U09 - 17	"READY"
!	P10 - 26B ->	U09 - 18	"ISSR"
!	U09 - 2 ->	U72 - 14, U63 - 4	"IRS"
!	U09 - 5 ->	U03 - 3	
!	U09 - 6 ->	U03 - 2	
!	U09 - 9 ->	U03 - 1	
!	U09 - 12 ->	U03 - 15	
!	U09 - 15 ->	U03 - 14	
!	U09 - 16 ->	U03 - 13	
!	U09 - 19 ->	U03 - 12	
!	VCC ->	U09 - 1	

! LATCH MICROCODE IN PIPELINE

```
TC273(POS,U01,RU[56],RU[55],U02,U03,RU[54],RU[4],
      U53,,K+32,ANCLK,U55,RU[2],RU[15],U42,U56,RU[1],RU[6],
      U51);
```

!	U60 - 2 ->	U18 - 3	"U01"
!	U60 - 5 ->	U18 - 13	"U02"
!	U60 - 6 ->	U25 - 3	"U03"
!	U60 - 9 ->	U21 - 7, U21 - 10	"U53"
!	U60 - 12 ->		"U55"
!	U60 - 15 ->	U64 - 5, U70 - 12	"U42"
!	U60 - 16 ->		"U56"
!	U60 - 19 ->	U01 - 5, U61 - 6	"U51"

TABLE 60 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

TC175(P05,U27,U27N,RU[30],RU[29],U28N,U28,.K+48,ANCLK,U29,
U29N,RU[28],RU[27],U30N,U30);

!	U67 - 2 ->	P10 - 48B	"U27"
!	U67 - 3 ->	P10 - 47B, U63 - 9	"U27*"
!	U67 - 6 ->	P10 - 43A, U56 - 2	"U28*"
!	U67 - 7 ->	P10 - 42A	"U28"
!	U67 - 10 ->	P10 - 41A	"U29"
!	U67 - 11 ->		"U29*"
!	U67 - 14 ->	P10 - 44B, U63 - 10, U56 - 3	"U30*"
!	U67 - 15 ->	P10 - 49B	"U30"

TC175(P05,U54,U54N,RU[3],RU[17],U40N,U40,.K+140,ANCLK,U41,
U41N,RU[16],RU[5],U52N,U52);

!	U68 - 2 ->		"U54"
!	U68 - 3 ->		"U54*"
!	U68 - 6 ->	U29 - 40, U32 - 40,	"U40*"
!		U35 - 40, U38 - 40	
!	U68 - 7 ->	U34 - 1, U70 - 14, U33 - 1	"U40"
!	U68 - 10 ->	U61 - 3, U70 - 13	"U41"
!	U68 - 11 ->	U45 - 19	"U41*"
!	U68 - 14 ->	U12 - 12	"U52*"
!	U68 - 15 ->	U61 - 10	"U52"

TC273(RPU,U37,RU[20],RU[19],U38,U39,RU[18],RU[48],
U09,.K+56,ANCLK,U07,RU[50],RU[49],U08,U25,
RU[32],RU[31],U26);

!	U02 - 2 ->	U03 - 9, U04 - 9	"U37"
!	U02 - 5 ->	U03 - 10, U04 - 10	"U38"
!	U02 - 6 ->	U03 - 11, U04 - 11	"U39"
!	U02 - 9 ->	U48 - 11,	"U09"
!	U02 - 12 ->	U11 - 2,	"U07"
!	U02 - 15 ->	U11 - 14,	"U08"
!	U02 - 16 ->	U64 - 9, U28 - 7	"U25"
!	U02 - 19 ->	U64 - 10, U28 - 1	"U26"
!	VCC ->	U02 - 1	

TABLE 60 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

TC273(RPU,U10,RU[47],RU[46],U11,U12,RU[45],RU[44],
U13,.K+72,ANCLK,U14,RU[43],RU[42],U15,U16,RU[41],
RU[39],U18);

!	U23 - 2 ->	U29 - 12, U32 - 12,	"U10"
!		U35 - 12, U38 - 12	
!	U23 - 5 ->	U29 - 13, U32 - 13,	"U11"
!		U35 - 13, U38 - 13	
!	U23 - 6 ->	U29 - 14, U32 - 14,	"U12"
!		U35 - 14, U38 - 14	
!	U23 - 9 ->	U29 - 26, U32 - 26,	"U13"
!		U35 - 26, U38 - 26	
!	U23 - 12 ->	U29 - 28, U32 - 28,	"U14"
!		U32 - 28, U38 - 28	
!	U23 - 15 ->	U29 - 27, U32 - 27,	"U15"
!		U32 - 27, U38 - 27	
!	U23 - 16 ->	U29 - 5, U32 - 5,	"U16"
!		U32 - 5, U38 - 5	
!	U23 - 19 ->	U29 - 6, U32 - 6,	"U18"
!		U35 - 6, U38 - 6	
!	VCC ->	U23 - 1	

TC175(RPU,U31,U31N,RU[26],RU[24],U33N,U33,.K+88,ANCLK,U36,
U36N,RU[21],RU[40],U17N,U17);

!	U10 - 2 ->	U07 - 4, U05 - 3; U65 - 5	"U31"
!	U10 - 3 ->	U07 - 1, U07 - 12, U27 - 10,	"U31*"
!		U64 - 2	
!	U10 - 6 ->	U05 - 5, U05 - 12, U14 - 15	"U33*"
!	U10 - 7 ->	U05 - 9, U07 - 14, U14 - 1	"U33"
!	U10 - 10 ->	U04 - 7	"U36"
!	U10 - 11 ->	U03 - 7	"U36*"
!	U10 - 14 ->	U11 - 15	"U17*"
!	U10 - 15 ->	U11 - 1, U29 - 7, U32 - 7,	"U17"
!		U35 - 7, U38 - 7	
!	VCC ->	U10 - 1	

TABLE 60 (CONT'D)

!	OUTPUT	!	DESTINATION	!	"SIG NAME"
---	--------	---	-------------	---	------------

```

TC273(RPU,U22,RU[35],RU[53],U04,U05,RU[52],RU[51],
      U06,.K+96,ANCLK,U19,RU[38],RU[25],U32,U34,RU[23],
      RU[22],U35);

```

!	U16 - 2 ->	U20 - 2, U49 - 12, U56 - 9	"U22"
!	U16 - 5 ->	U27 - 1, U27 - 4	"U04"
!	U16 - 6 ->	U18 - 2, U25 - 2	"U05"
!	U16 - 9 ->	U18 - 14, U25 - 14	"U06"
!	U16 - 12 ->	U20 - 10	"U19"
!	U16 - 15 ->	U05 - 2, U05 - 6, U07 - 2	"U32"
!	U16 - 16 ->	U14 - 2	"U34"
!	U16 - 19 ->	U14 - 14	"U35"
!	VCC ->	U16 - 1	

```

TC175(RPU,U20,U20N,RU[37],RU[36],U21N,U21,.K+108,ANCLK,
      U23,U23N,RU[34],RU[33],U24N,U24);

```

!	U17 - 2 ->	U50 - 2, U51 - 2, U57 - 2,	"U20"
!		U58 - 2	
!	U17 - 3 ->	U49 - 9, U56 - 11, U65 - 10,	"U20*"
!		U19 - 12	
!	U17 - 6 ->	U27 - 12, U56 - 12	"U21*"
!	U17 - 7 ->	U19 - 9, U20 - 1, U56 - 8	"U21"
!	U17 - 10 ->	U19 - 10, U65 - 1	"U23"
!	U17 - 11 ->	U56 - 5, U64 - 13	"U23*"
!	U17 - 14 ->	U56 - 6, U65 - 2	"U24*"
!	U17 - 15 ->	U27 - 13, U65 - 9, U58 - 13	"U24"
!	VCC ->	U17 - 1	
!	P10 - 50B ->	U60 - 1, U67 - 1, U68 - 1	"POS"

```

END; ! ROUTINE PARTN3
END
ELUDOM ! P3

```

BEGIN

```
external routine tc153,tc253,tc2901a,taluout;
```

) 8

LOCAL A, B, C, RAM34;

```
!      OUTPUT          DESTINATION          "SIG NAME"
!
```

```
tc153(F,U05,U03,IR09,IR01,IR05,SPB01,GND,TEMP,IR04,IR00,  
IR08,F,U06,F);
```

223

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

```
IC153(F,U05,U01,F,IR03,IR07,SPB03,GND,SPB02,IR06,IR02,F,
U02,U06,F);
```

!	U18 = 7 ->	U12 = 9, U29 = 20, U32 = 20,	"SPB03"
!		U35 = 20, U38 = 20	
!	U18 = 9 ->	U72 = 9, U29 = 19, U32 = 19,	"SPB02"
!		U35 = 19, U38 = 19	
!	GND ->	U18 = 1, U18 = 4, U18 = 12,	
!		U18 = 15	

```
ORR(SPBO0,TEMP,U04);
```

!	U27 = 6 ->	U72 = 11, U29 = 17, U32 = 17,	"SPBO"
!		U35 = 17, U38 = 17	

```
!      MAIN PROCESSOR - RUN FOUR 2901AS IN PARALLELS. THE
!      2902 IS SIMULATED EXPLICITLY AS GATES BETWEEN
!      2901A INVOCATIONS. FINALLY, THE 2901A IS EVALUATED
!      IN TWO PARTS - ONE CALCULATES THE OUTPUT LINE STATE
!      AND ONE CALCULATES THE CHANGE IN INTERNAL STATE OF
!      THE CHIP. THIS IS DONE TO INSURE PROPER SHIFTING
!      OPERATION.
```

```
ORR(SPA0,IR00,U04);
```

!	U27 = 3 ->	U29 = 4, U32 = 4, U35 = 4,	"SPA0"
!		U38 = 4	

```
INV(CN,U09); ZERO = -1;
```

!	U48 = 10 ->	U24 = 13, U38 = 29	"CN"
---	-------------	--------------------	------

TABLE 61 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

```
TC2901A(IR03,IR02,IR01,SPA0,U16,U18,U17,RAM31,RAM01,
J,ZERO,U10,U11,U12,ANCLK,Q31,SPB00,SPB01,SPB02,SPB03,
Q01,D03,D02,D01,D00,U13,U15,U14,CN,0,XX1,GO,XX1,
XX1,P0,YN00,YN01,YN02,YN03,U40N);
```

!	U24 - 3 ->	U38 - 32	
!	U24 - 4 ->	U38 - 35	
!	U38 - 36 ->	U31 - 18, U34 - 19, U46 - 14	"Y00"
!		U58 - 12, U28 - 3	
!	U38 - 37 ->	U31 - 17, U34 - 16, U46 - 13	"Y01"
!		U58 - 4, U28 - 4	
!	U38 - 38 ->	U31 - 14, U34 - 15, U46 - 8	"Y02"
!		U51 - 12, U28 - 5	
!	U38 - 39 ->	U31 - 13, U34 - 12, U46 - 7	"Y03"
!		U51 - 4, U28 - 6	

```
INV(CNBAR,CN);      A = ANDF(P0,GO);
```

```
B = ANDF(GO,CNBAR);  NOR(CNX,A,B);
```

TABLE 61 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

```

TC2901A(IR03,IR02,IR01,SPA0,U16,U18,U17,RAM32,RAM02,
J,ZERO,U10,U11,U12,ANCLK,Q32,SPB00,SPB01,SPB02,SPB03,
Q02,D07,D06,D05,D04,U13,U15,U14,CNX,130,XX1,G1,XX1,
XX1,P1,YN04,YN05,YN06,YN07,U40N);

```

!	U24 - 1 ->	U32 - 32	
!	U24 - 2 ->	U32 - 35	
!	U24 - 12 ->	U32 - 29	
!	U38 - 16 ->	U32 - 21	
!	U38 - 8 ->	U32 - 9	
!	U32 - 36 ->	U31 - 8, U34 - 9, U46 - 14,	"Y04"
!		U57 - 12, U21 - 3	
!	U32 - 37 ->	U31 - 7, U34 - 6, U46 - 13,	"Y05"
!		U57 - 4, U21 - 4	
!	U32 - 38 ->	U31 - 4, U34 - 4, U46 - 8,	"Y06"
!		U50 - 12, U21 - 5	
!	U32 - 39 ->	U31 - 3, U34 - 2, U46 - 7,	"Y07"
!		U50 - 4, U21 - 6	

A = ANDF(P1,G1); B = AND3(P0,G1,G0);

C = AND3(CNBAR,G0,G1); NOR3(CNY,A,B,C);

TABLE 61 (CONT'D)

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

```

TC2901A(IR03,IR02,IR01,SPA0,U16,U16,U17,RAM33,RAM03,
J,ZERO,U10,U11,U12,ANCLK,Q33,SPB00,SPB01,SPB02,SPB03,
Q03,D11,D10,D09,D08,U13,U15,U14,CNY,260,XX1,G2,XX1,
XX1,P2,YN08,YN09,YN10,YN11,U40N);

```

!	U24 - 11 ->	U29 - 29	
!	U24 - 14 ->	U29 - 32	
!	U24 - 15 ->	U29 - 35	
!	U32 - 8 ->	U29 - 9	
!	U32 - 16 ->	U29 - 21	
!	U29 - 36 ->	U30 - 18, U33 - 19, U47 - 13	"Y08"
!		U45 - 1, U46 - 4	
!	U29 - 37 ->	U30 - 17, U33 - 16, U47 - 10	"Y09"
!		U45 - 2, U46 - 3	
!	U29 - 29 ->	U30 - 14, U33 - 15, U47 - 6	"Y10"
!		U45 - 3	
!	U29 - 39 ->	U30 - 13, U33 - 12, U47 - 3	"Y11"
!		U45 - 4	

A = ANDF(P2,G2); B = ANDJ(P1,G2,G1);

C = AND4F(P0,G2,G1,G0); TEMP = AND4F(G2,G1,G0,CNBAR);

TABLE 61 (CONT'D)

```
NOR4(CNZ,A,B,C,TEMP);
```

```
IC2901A(IR03,IR02,IR01,SPA0,U16,U18,U17,RAM34,RAM04,
J,ZERO,U10,U11,U12,ANCLK,Q34,SPB00,SPB01,SPB02,SPB03,
Q04,D15,D14,D13,D12,U13,U15,U14,CNZ,390,SIGNV,XX1,COUT,
AOV,XX2,YN12,YN13,YN14,YN15,U40N);
```

```
!      U24 = 9  ->    U35 = 29
!      U29 = 8  ->    U35 = 9
!      U29 = 16 ->    U35 = 21
!      U35 = 36 ->    U30 = 8, U33 = 9, U45 = 5,  "Y12"
!                      U54 = 13
!      U35 = 37 ->    U30 = 7, U33 = 6, U45 = 16, "Y13"
!                      U54 = 10
!      U35 = 38 ->    U30 = 4, U33 = 5, U45 = 17, "Y14"
!                      U54 = 6, U12 = 2
!      U35 = 39 ->    U30 = 3, U33 = 2, U45 = 18, "Y15"
!                      U54 = 3, U12 = 1, U14 = 3
```

! DETERMINE PROPER BITS FOR SHIFT OPERATIONS

```
XORR(QBIT,IND,SIGNV);
```

```
!      U12 = 6  ->    U11 = 12, U14 = 13                "QBIT"
```

```
XORR(TEMP,SIGNV,AOV);
```

```
!      U44 = 3  ->    U11 = 5
```

```
TC253(U17,U07,SQ00,IND,TEMP,F,SRAM15,GND,SQ00,F,SIGNV,
QBIT,T,U08,U17N);
```

```
!      U11 = 7  ->    U35 = 8                            "SRAM15"
!      U11 = 9  ->    U11 = 3, U04 = 15, U38 = 21       "SQ00"
!      GND       ->    U11 = 6, U11 = 10
```

!	OUTPUT	DESTINATION	"SIG NAME"
---	--------	-------------	------------

!	EVALUTING BIDIRECTIONAL LINES BY ORING TOGETHER ALL		
!	OUTPUTS ON EACH LINE.		

SQ00 = .SQ00 AND .Q01; Q02 = .Q02 AND .Q31; Q03 = .Q03 AND .Q32;
 Q04 = .Q04 AND .Q33; SRAM = .Q34 AND .RAM01;
 RAM02 = .RAM02 AND .RAM31; RAM03 = .RAM03 AND .RAM32;
 RAM04 = .RAM04 AND .RAM33; SRAM15 = .SRAM15 AND .RAM34;

!	EVALUATE THE INTERNAL STATES OF THE CHIP (Q REG
!	AND RAM) NOW THAT THE PROPER VALUES ARE ON THE
!	CARRY AND SHIFT LINE

TALUOUT(SQ00,Q02,SRAM,RAM02,0);

TALUOUT(Q02,Q03,RAM02,RAM03,130);

TALUOUT(Q03,Q04,RAM03,RAM04,260);

TALUOUT(Q04,SRAM,RAM04,SRAM15,390);

END;

END ELUDOM

TABLE 61 (CONT'D)

CPU CARD CHIP INTERCONNECTIONS

OUTPUT	DESTINATION	"SIG NAME"
U20 - 3	-> U50 - 14, U51 - 14, U57 - 14, U58 - 14	"FMEA3"
U58 - 7	-> U43 - 19	"DD01"
U58 - 9	-> U43 - 21	"DD00"
VCC	-> U58 - 3, U58 - 13	
GND	-> U58 - 15	
U51 - 7	-> U43 - 15	"DD03"
U51 - 9	-> U43 - 17	"DD02"
VCC	-> U51 - 3, U51 - 13	
U57 - 7	-> U42 - 19	"DD11"
U57 - 9	-> U42 - 21	"DD10"
VCC	-> U57 - 3, U57 - 13	
U50 - 7	-> U42 - 15, U49 - 10	"DD13"
U50 - 9	-> U42 - 17	"DD12"
VCC	-> U50 - 3, U50 - 13	
U56 - 13	-> U54 - 1, U47 - 1	"FMEA1"
U49 - 8	-> U49 - 13	
U49 - 11	-> U54 - 15, U47 - 15	"FMEA2"
U54 - 4	-> U39 - 15	"DD33"
U54 - 7	-> U39 - 17	"DD32"
U54 - 9	-> U39 - 19	"DD31"
U54 - 12	-> U39 - 21	"DD30"
VCC	-> U54 - 2, U54 - 5, U54 - 11, U54 - 14	
U47 - 4	-> U40 - 15	"DD23"
U47 - 7	-> U40 - 17	"DD22"
U47 - 9	-> U40 - 19	"DD21"
U47 - 12	-> U40 - 21	"DD20"
VCC	-> U47 - 2, U47 - 5, U47 - 11, U47 - 14	
U19 - 8	-> U39 - 5, U40 - 5, U42 - 5, U43 - 5, P10 - 14A	"13"
U27 - 11	-> U64 - 12	
U64 - 11	-> U39 - 3, U40 - 3, U42 - 3, U43 - 3, P10 - 40A	"11"
U41 - 4	-> U39 - 7, U17 - 9, U29 - 15, U40 - 7, U16 - 11, U32 - 15, U42 - 7, U10 - 9, U35 - 15, U43 - 7, U23 - 11, U38 - 15, U2 - 11, U53 - 11, U60 - 11, U9 - 11, U67 - 9, U68 - 9, U28 - 2, U21 - 2, U46 - 11, P09 - 30B	"A"
GND	-> U41 - 6	
P10 - 8B	-> U39 - 22, U40 - 22, U42 - 22, U43 - 22	"QMAR"
U43 - 13	-> U42 - 23	
U43 - 14	-> P10 - 4B	"MAR03"
U43 - 16	-> P10 - 5B	"MAR02"
U43 - 18	-> P10 - 6B	"MAR01"
U43 - 20	-> P10 - 7B	"MAR00"
VCC	-> U43 - 23	
U42 - 13	-> U40 - 23	
U42 - 14	-> P10 - 13B	"MAR07"
U42 - 16	-> P10 - 15B	"MAR06"

U42 - 18	->	P10 - 16B	"MAR05**"
U42 - 20	->	P10 - 17B	"MAR04**"
U40 - 13	->	U39 - 23	
U40 - 14	->	P10 - 35B	"MAR11**"
U40 - 16	->	P10 - 36B	"MAR10**"
U40 - 18	->	P10 - 38B	"MAR09**"
U40 - 20	->	P10 - 39B	"MAR08**"
U39 - 14	->	P10 - 52B	"MAR15**"
U39 - 16	->	P10 - 53B	"MAR14**"
U39 - 18	->	P10 - 54B	"MAR13**"
U39 - 20	->	P10 - 56B	"MAR12**"
U56 - 10	->	U50 - 1, U50 - 15, U65 - 11, U51 - 1, U51 - 15, U57 - 1, U57 - 15, U58 - 1, U19 - 13	"E7DI**"
U19 - 11	->	U39 - 1, U40 - 1, U42 - 1, U43 - 1, U65 - 13, P10 - 7A,	"EX**"
U56 - 4	->	U39 - 4, U40 - 4, U42 - 4, U43 - 4, P10 - 16A,	"I2"
U65 - 8	->	U36 - 1, U37 - 1, U48 - 5	"EDR**"
U48 - 6	->	U39 - 6, U40 - 6, U42 - 6, U43 - 6, P10 - 26A	"EDR"
U37 - 2	->	U32 - 22, U42 - 11	"D07"
U37 - 5	->	U32 - 23, U42 - 10	"D06"
U37 - 6	->	U32 - 24, U42 - 09	"D05"
U37 - 9	->	U32 - 25, U42 - 08	"D04"
U37 - 12	->	U38 - 22, U43 - 11	"D03"
U37 - 15	->	U38 - 23, U43 - 10	"D02"
U37 - 16	->	U38 - 24, U43 - 09	"D01"
U37 - 19	->	U38 - 25, U43 - 08	"D00"
U36 - 2	->	U35 - 22, U39 - 11	"D15"
U36 - 5	->	U35 - 23, U39 - 10	"D14"
U36 - 6	->	U35 - 24, U39 - 9	"D13"
U36 - 9	->	U35 - 25, U39 - 8	"D12"
U36 - 12	->	U29 - 22, U40 - 11	"D12"
U36 - 15	->	U29 - 23, U40 - 10	"D10"
U36 - 16	->	U29 - 24, U40 - 9	"D09"
U36 - 19	->	U29 - 25, U40 - 8	"D08"
U56 - 1	->	U20 - 4	
P10 - 17A	->	U63 - 12	
U63 - 8	->	U20 - 13	
VCC	->	U63 - 13	
P10 - 11B	->	U20 - 5, U20 - 12, U41 - 5, U65 - 4	"A1"
U20 - 6	->	U33 - 11, U34 - 11, P10 - 20B	"TIB**"
U20 - 8	->	U30 - 11, U31 - 11, P10 - 18B	"TOR**"
U20 - 11	->	U36 - 11, U37 - 11, P10 - 19B	"TOR**"
P10 - 29B	->	U30 - 1, U31 - 1	
U31 - 2	->	U34 - 3, U37 - 3, U71 - 12, P10 - 13A	"DAT07"
U31 - 5	->	U34 - 4, U37 - 4, U71 - 11, P10 - 39A	"DAT06"
U31 - 6	->	U34 - 7, U37 - 7, U71 - 2, P10 - 12A	"DAT05"
U31 - 9	->	U34 - 8, U37 - 8, U71 - 3, P10 - 11A	"DA104"
U31 - 12	->	U34 - 13, U37 - 13, U19 - 1, P10 - 21B	"DA103"
U31 - 15	->	U34 - 14, U37 - 14, U27 - 9, P10 - 22B	"DAT02"
U31 - 16	->	U34 - 17, U37 - 17, U07 - 5, P10 - 9A	"DAT01"

U31 - 19	->	U34 - 18, U37 - 18, U07 - 11,	"DAT00"
		P10 - 27B	
U30 - 2	->	U33 - 3, U36 - 3, P10 - 38A	"DAT15"
U30 - 5	->	U33 - 4, U36 - 4, P10 - 36A	"DAT14"
U30 - 6	->	U33 - 7, U36 - 7, P10 - 33A	"DAT13"
U30 - 9	->	U33 - 8, U36 - 8, P10 - 31A	"DAT12"
U30 - 12	->	U33 - 13, U36 - 13, P10 - 31B	"DAT11"
U30 - 15	->	U33 - 14, U36 - 14, P10 - 34A	"DAT10"
U30 - 16	->	U33 - 17, U36 - 17, P10 - 33B	"DAT09"
U30 - 19	->	U33 - 18, U36 - 18, P10 - 34B	"DAT08"
U38 - 9	->	U14 - 10, U35 - 16	"SRAM00"
U14 - 7	->	U05 - 8, U13 - 2	
U14 - 9	->	U05 - 11, U13 - 12	
VCC	->	U14 - 4, U14 - 12	
GND	->	U14 - 5, U14 - 11	
U5 - 10	->	U13 - 3	
U5 - 13	->	U13 - 11	
P10 - 23B	->	U13 - 1, U13 - 13, U6 - 13,	"A"
		U06 - 1	
U13 - 6	->	U04 - 3, U11 - 4, U12 - 4,	
		U14 - 6, U44 - 5, P10 - 1A	
U13 - 5	->	P10 - 3B	"IND*"
U13 - 8	->	U04 - 2, P10 - 2B	"LINK"
VCC	->	U13 - 4, U13 - 10	
U12 - 3	->	U7 - 3	
U35 - 34	->	U48 - 1, U7 - 6, U44 - 2,	"AOV"
		U9 - 4, P10 - 30A	
U48 - 2	->	U7 - 10	
U05 - 1	->	U7 - 15	
U07 - 7	->	U06 - 2	
U07 - 9	->	U06 - 3	
GND	->	U07 - 13	
U05 - 4	->	U19 - 2, U19 - 4, U64 - 1,	
		U65 - 3	
U27 - 8	->	U19 - 5	
U19 - 3	->	U06 - 12	
U19 - 6	->	U06 - 11	
U64 - 3	->	U62 - 1	
U06 - 5	->	U04 - 1	"FOV*"
U06 - 6	->	U72 - 15, P10 - 35A	"FOV"
U06 - 8	->	U63 - 5, U72 - 1, P10 - 5A	"PFEI*"
U06 - 9	->	U41 - 12	"NPFEI*"
VCC	->	U06 - 4, U06 - 10	
U65 - 6	->	U71 - 1, U71 - 13	
U62 - 3	->	P10 - 27A	"IAM*"
GND	->	U62 - 2	
U71 - 6	->	U72 - 13, P10 - 8A	"FLAG1"
U71 - 8	->	U72 - 12, P10 - 10A	"FLAG2"
VCC	->	U71 - 4, U71 - 10	
U64 - 5	->	U28 - 9	
GND	->	U28 - 10	
U28 - 15	->	U04 - 14, U61 - 4,	"RPTOV*"
		P10 - 37A	
U72 - 6	->	U12 - 10	
GND	->	U72 - 7	
P10 - 2A	->	U72 - 4	"EXT1"
P10 - 3A	->	U72 - 3	"EXT2"
P10 - 4A	->	U72 - 2	"EXT3"
U12 - 6	->	U09 - 14	"CON"
U65 - 12	->	P10 - 6A	"HLTLP"
P10 - 24B	->	U04 - 13	"SAT*"

TABLE 62 (CONT'D)

U04 - 6	->	U49 - 1		
VCC	->	U04 - 4		
U03 - 6	->	U49 - 2		
U49 - 3	->	U70 - 11, U12 - 13,	"COND"	
		P10 - 18A		
U64 - 6	->	U49 - 4, U61 - 13	"HALT*"	
P10 - 24A	->	U63 - 1	"ARM"	
U63 - 6	->	U49 - 5		
U49 - 6	->	U70 - 10		
P10 - 19A	->	U46 - 13	"TEST*"	
U48 - 12	->	U70 - 15		
U70 - 1	->	U53 - 1, P10 - 23A	"EBCH*"	
U70 - 2	->	U45 - 15, P10 - 22A	"ESTRT*"	
U70 - 3	->	U69 - 1, U69 - 15, P10 - 21A	"ESPC*"	
U70 - 4	->	U61 - 1, U61 - 15, P10 - 20A	"ELSB*"	
U70 - 5	->	U62 - 4, U62 - 10	"EMSB*"	
U70 - 6	->	U61 - 2	"SB"	
U70 - 7	->	U61 - 14, U41 - 2	"SA"	
U70 - 9	->	U46 - 1, U21 - 9, P10 - 15A	"ELIR*"	
U12 - 11	->	U61 - 11, U61 - 12		
U61 - 7	->	U45 - 7, U15 - 2, U52 - 2,	"UMA1"	
		U08 - 2, U59 - 2, U22 - 2,		
		U66 - 2, U01 - 2, P10 - 54A		
U61 - 9	->	U45 - 6, U15 - 1, U52 - 1,	"UMA0"	
		U08 - 1, U59 - 1, U22 - 1,		
		U66 - 1, U01 - 1, P10 - 55A		
GND	->	U69 - 2, U69 - 4, U69 - 6, U69 - 10,		
		U69 - 12, U69 - 14		
U69 - 9	->	U45 - 8, U15 - 3, U52 - 3,	"UMA2"	
		U08 - 3, U59 - 3, U22 - 3,		
		U66 - 3, U01 - 3, U53 - 19,		
		P10 - 53A		
U69 - 7	->	U45 - 9, U15 - 4, U52 - 4,	"UMA3"	
		U08 - 4, U59 - 4, U22 - 4,		
		U66 - 4, U01 - 4, U53 - 16,		
		P10 - 52A		
U69 - 5	->	U45 - 11, U15 - 5, U52 - 5,	"UMA4"	
		U08 - 5, U59 - 5, U22 - 5,		
		U66 - 5, U01 - 5, U53 - 15,		
		P10 - 51A		
U69 - 3	->	U45 - 12, U15 - 16, U52 - 16,	"UMA5"	
		U08 - 16, U59 - 16, U22 - 16,		
		U12 - 16, U66 - 16, U53 - 12,		
		P10 - 50A		
U69 - 11	->	U45 - 13, U15 - 17, U52 - 17,	"UMA6"	
		U08 - 17, U59 - 17, U22 - 17,		
		U66 - 17, U01 - 17, U53 - 9,		
		P10 - 49A		
U69 - 13	->	U45 - 14, U15 - 18, U52 - 18,	"UMA7"	
		U08 - 18, U59 - 18, U22 - 18,		
		U66 - 18, U01 - 18, U53 - 6,		
		P10 - 48A		
GND	->	U62 - 5, U62 - 9		
U62 - 8	->	U15 - 19, U52 - 19, U53 - 5,	"UMA8"	
		U08 - 19, U22 - 19, U59 - 19,		
		U66 - 19, U01 - 19, P10 - 47A		
U62 - 6	->	U15 - 15, U52 - 15, U53 - 2,	"UMA9"	
		U08 - 15, U22 - 15, U59 - 15,		
		U66 - 15, U01 - 15, P10 - 46A		
U52 - 6	->	U68 - 13		
U52 - 7	->	U60 - 18		

TABLE 62 (CONT'D)

U52 - 8	->	U53 - 7
U52 - 9	->	U53 - 8
U52 - 11	->	U53 - 13
U52 - 12	->	U53 - 14
U52 - 13	->	U53 - 17
U52 - 14	->	U53 - 18
U59 - 6	->	U60 - 13
U59 - 7	->	U60 - 17
U59 - 8	->	U60 - 3
U59 - 9	->	U60 - 4
U59 - 11	->	U60 - 7
U59 - 12	->	U60 - 8
U59 - 13	->	U53 - 3
U59 - 14	->	U53 - 4
U66 - 6	->	U68 - 12
U66 - 7	->	U60 - 14
U66 - 8	->	U67 - 4
U66 - 9	->	U67 - 5
U66 - 11	->	U67 - 12
U66 - 12	->	U67 - 13
U66 - 13	->	U68 - 4
U66 - 14	->	U68 - 5
U01 - 6	->	U02 - 3
U01 - 7	->	U02 - 4
U01 - 8	->	U02 - 7
U01 - 9	->	U02 - 8
U01 - 11	->	U02 - 13
U01 - 12	->	U02 - 14
U01 - 13	->	U02 - 17
U01 - 14	->	U02 - 18
U22 - 6	->	U23 - 3
U22 - 7	->	U23 - 4
U22 - 8	->	U23 - 7
U22 - 9	->	U23 - 8
U22 - 11	->	U23 - 13
U22 - 12	->	U23 - 14
U22 - 13	->	U23 - 17
U22 - 14	->	U23 - 18
U08 - 6	->	U16 - 13
U08 - 7	->	U16 - 14
U08 - 8	->	U16 - 17
U08 - 9	->	U16 - 18
U08 - 11	->	U10 - 4
U08 - 12	->	U10 - 5
U08 - 13	->	U10 - 12
U08 - 14	->	U10 - 13
U15 - 6	->	U17 - 4
U15 - 7	->	U17 - 5
U15 - 8	->	U17 - 12
U15 - 9	->	U17 - 13
U15 - 11	->	U16 - 3
U15 - 12	->	U16 - 4
U15 - 13	->	U16 - 7
U15 - 14	->	U16 - 8
U21 - 11	->	U50 - 5, U18 - 6
U21 - 12	->	U50 - 11, U18 - 10
U21 - 13	->	U57 - 5, U25 - 6
U21 - 14	->	U57 - 11, U25 - 10
VCC	->	U21 - 1
P10 - 25A	->	U48 - 9
U41 - 1	->	U41 - 11

"IR07"
 "IR06"
 "IR05"
 "IR04"

"HALT*"

U41 - 13	->	U46 - 17	
U48 - 6	->	U46 - 18	
U46 - 2	->	U25 - 4, P10 - 45B	"IR09"
U46 - 5	->	U25 - 12, P10 - 40B	"IR08"
U46 - 6	->	U18 - 5, U29 - 1,	"IR03"
		U32 - 1, U35 - 1,	
		U38 - 1, U51 - 5, U51 - 6,	
		U57 - 6, U57 - 10, U50 - 6,	
		U50 - 10, P10 - 51B	
U46 - 9	->	U18 - 11, U51 - 10,	"IR02"
		U51 - 11, U29 - 2,	
		U32 - 2, U35 - 2, U38 - 2,	
		P10 - 43B	
U46 - 12	->	U25 - 5, U58 - 5,	"IR01"
		U58 - 6, U29 - 3,	
		U32 - 3, U35 - 3, U38 - 3,	
		P10 - 12B	
U46 - 15	->	U25 - 11, U58 - 10,	"IR00"
		U58 - 11, U27 - 2,	
		P10 - 10B	
U46 - 16	->	U63 - 2, U41 - 3	"FEIN"
U46 - 19	->	U64 - 4	"HALTS"
P10 - 29A	->	U48 - 3	"IR**"
U48 - 4	->	U09 - 3	
U35 - 11	->	U09 - 7, U29 - 11, U32 - 11	"ZERO"
		U38 - 11, P10 - 9B	
U35 - 31	->	U09 - 8, U11 - 11,	"SIGN"
		U44 - 1, U12 - 5, P10 - 41B	
U35 - 33	->	U09 - 13, P10 - 42B	"COUT"
P10 - 25B	->	U09 - 17	"READY"
P10 - 26B	->	U09 - 18	"TSSR"
U09 - 2	->	U72 - 14, U63 - 4	"IRS"
U09 - 5	->	U03 - 3	
U09 - 6	->	U03 - 2	
U09 - 9	->	U03 - 1	
U09 - 12	->	U03 - 15	
U09 - 15	->	U03 - 14	
U09 - 16	->	U03 - 13	
U09 - 19	->	U03 - 12	
VCC	->	U09 - 1	
U60 - 2	->	U18 - 3	"U01"
U60 - 5	->	U18 - 13	"U02"
U60 - 6	->	U25 - 3	"U03"
U60 - 9	->	U21 - 7, U21 - 10	"U53"
U60 - 12	->		"U55"
U60 - 15	->	U64 - 5, U70 - 12	"U42"
U60 - 16	->		"U56"
U60 - 19	->	U61 - 5, U61 - 6	"U51"
U67 - 2	->	P10 - 48B	"U27"
U67 - 3	->	P10 - 47B, U63 - 9	"U27*"
U67 - 6	->	P10 - 43A, U56 - 2	"U26*"
U67 - 7	->	P10 - 42A	"U28"
U67 - 10	->	P10 - 41A	"U29"
U67 - 11	->		"U29*"
U67 - 14	->	P10 - 44B, U63 - 10, U56 - 3	"U30*"
U67 - 15	->	P10 - 49B	"U30"
U68 - 2	->		"U54"
U68 - 3	->		"U54*"
U68 - 6	->	U29 - 40, U32 - 40,	"U40*"
		U35 - 40, U38 - 40	
U68 - 7	->	U34 - 1, U70 - 14, U33 - 1	"U40"

U68 - 10	-> U61 - 3, U70 - 13	"U41"
U68 - 11	-> U45 - 19	"U41*"
U68 - 14	-> U12 - 12	"U52*"
U68 - 15	-> U61 - 10	"U52"
U02 - 2	-> U03 - 9, U04 - 9	"U37"
U02 - 5	-> U03 - 10, U04 - 10	"U38"
U02 - 6	-> U03 - 11, U04 - 11	"U39"
U02 - 9	-> U48 - 11,	"U09"
U02 - 12	-> U11 - 2,	"U07"
U02 - 15	-> U11 - 14,	"U08"
U02 - 16	-> U64 - 9, U28 - 7	"U25"
U02 - 19	-> U64 - 10, U28 - 1	"U26"
VCC	-> U02 - 1	
U23 - 2	-> U29 - 12, U32 - 12,	"U10"
	U35 - 12, U38 - 12	
U23 - 5	-> U29 - 13, U32 - 13,	"U11"
	U35 - 13, U38 - 13	
U23 - 6	-> U29 - 14, U32 - 14,	"U12"
	U35 - 14, U38 - 14	
U23 - 9	-> U29 - 26, U32 - 26,	"U13"
	U35 - 26, U38 - 26	
U23 - 12	-> U29 - 28, U32 - 28,	"U14"
	U32 - 28, U38 - 28	
U23 - 15	-> U29 - 27, U32 - 27,	"U15"
	U32 - 27, U38 - 27	
U23 - 16	-> U29 - 5, U32 - 5,	"U16"
	U32 - 5, U38 - 5	
U23 - 19	-> U29 - 6, U32 - 6,	"U18"
	U35 - 6, U38 - 6	
VCC	-> U23 - 1	
U10 - 2	-> U07 - 4, U05 - 3, U65 - 5	"U31"
U10 - 3	-> U07 - 1, U07 - 12, U27 - 10,	"U31*"
	U64 - 2	
U10 - 6	-> U05 - 5, U05 - 12, U14 - 15	"U33*"
U10 - 7	-> U05 - 9, U07 - 14, U14 - 1	"U33"
U10 - 10	-> U04 - 7	"U36"
U10 - 11	-> U03 - 7	"U36*"
U10 - 14	-> U11 - 15	"U17*"
U10 - 15	-> U11 - 1, U29 - 7, U32 - 7,	"U17"
	U35 - 7, U38 - 7	
VCC	-> U10 - 1	
U16 - 2	-> U20 - 2, U49 - 12, U56 - 9	"U22"
U16 - 5	-> U27 - 1, U27 - 4	"U04"
U16 - 6	-> U18 - 2, U25 - 2	"U05"
U16 - 9	-> U18 - 14, U25 - 14	"U06"
U16 - 12	-> U20 - 10	"U19"
U16 - 15	-> U05 - 2, U05 - 6, U07 - 2	"U32"
U16 - 16	-> U14 - 2	"U34"
U16 - 19	-> U14 - 14	"U35"
VCC	-> U16 - 1	
U17 - 2	-> U50 - 2, U51 - 2, U57 - 2,	"U20"
	U58 - 2	
U17 - 3	-> U49 - 9, U56 - 11, U65 - 10,	"U20*"
	U19 - 12	
U17 - 6	-> U27 - 12, U56 - 12	"U21*"
U17 - 7	-> U19 - 9, U20 - 1, U56 - 8	"U21"
U17 - 10	-> U19 - 10, U65 - 1	"U23"
U17 - 11	-> U56 - 5, U64 - 13	"U23*"
U17 - 14	-> U56 - 6, U65 - 2	"U24*"
U17 - 15	-> U27 - 13, U65 - 9, U58 - 13	"U24"
VCC	-> U17 - 1	

TABLE 62 (CONT'D)

P10 - 50B	->	U60 - 1, U67 - 1, U68 - 1	"POS"
U25 - 7	->	U72 - 10, U29 - 18, U32 - 18,	"SPB1"
		U35 - 18, U38 - 18	
U25 - 9	->	U27 - 5	
GND	->	U25 - 1, U25 - 13, U25 - 15	
U18 - 7	->	U12 - 9, U29 - 20, U32 - 20,	"SPB03"
		U35 - 20, U38 - 20	
U18 - 9	->	U72 - 9, U29 - 19, U32 - 19,	"SPB02"
		U35 - 19, U38 - 19	
GND	->	U18 - 1, U18 - 4, U18 - 12,	
		U18 - 15	
U27 - 6	->	U72 - 11, U29 - 17, U32 - 17,	"SPB0"
		U35 - 17, U38 - 17	
U27 - 3	->	U29 - 4, U32 - 4, U35 - 4,	"SPA0"
		U38 - 4	
U48 - 10	->	U24 - 13, U38 - 29	"CN"
U24 - 3	->	U38 - 32	
U24 - 4	->	U38 - 35	
U38 - 36	->	U31 - 18, U34 - 19, U46 - 14	"Y00"
		U58 - 12, U28 - 3	
U38 - 37	->	U31 - 17, U34 - 16, U46 - 13	"Y01"
		U58 - 4, U28 - 4	
U38 - 38	->	U31 - 14, U34 - 15, U46 - 8	"Y02"
		U51 - 12, U28 - 5	
U38 - 39	->	U31 - 13, U34 - 12, U46 - 7	"Y03"
		U51 - 4, U28 - 6	
U24 - 1	->	U32 - 32	
U24 - 2	->	U32 - 35	
U24 - 12	->	U32 - 29	
U38 - 16	->	U32 - 21	
U38 - 8	->	U32 - 9	
U32 - 36	->	U31 - 8, U34 - 9, U46 - 14,	"Y04"
		U57 - 12, U21 - 3	
U32 - 37	->	U31 - 7, U34 - 6, U46 - 13,	"Y05"
		U57 - 4, U21 - 4	
U32 - 38	->	U31 - 4, U34 - 4, U46 - 8,	"Y06"
		U50 - 12, U21 - 5	
U32 - 39	->	U31 - 3, U34 - 2, U46 - 7,	"Y07"
		U50 - 4, U21 - 6	
U24 - 11	->	U29 - 29	
U24 - 14	->	U29 - 32	
U24 - 15	->	U29 - 35	
U32 - 8	->	U29 - 9	
U32 - 16	->	U29 - 21	
U29 - 30	->	U30 - 18, U33 - 19, U47 - 13	"Y08"
		U45 - 1, U46 - 4	
U29 - 37	->	U30 - 17, U33 - 16, U47 - 10	"Y09"
		U45 - 2, U46 - 3	
U29 - 29	->	U30 - 14, U33 - 15, U47 - 6	"Y10"
		U45 - 3	
U29 - 39	->	U30 - 13, U33 - 12, U47 - 3	"Y11"
		U45 - 4	
U24 - 9	->	U35 - 29	
U29 - 8	->	U35 - 9	
U29 - 16	->	U35 - 21	
U35 - 36	->	U30 - 8, U33 - 9, U45 - 5,	"Y12"
		U54 - 13	
U35 - 37	->	U30 - 7, U33 - 6, U45 - 16,	"Y13"
		U54 - 10	
U35 - 38	->	U30 - 4, U33 - 5, U45 - 17,	"Y14"
		U54 - 6, U12 - 2	

TABLE 62 (CONT'D)

U35 - 39	->	U30 - 3, U33 - 2, U45 - 18,	"Y15"
		U54 - 3, U12 - 1, U14 - 3	
U12 - 6	->	U11 - 12, U14 - 13	"QB17"
U44 - 3	->	U11 - 5	
U11 - 7	->	U35 - 8	"SRAM15"
U11 - 9	->	U11 - 3, U04 - 15, U38 - 21	"SQ00"
GND	->	U11 - 6, U11 - 10	

TABLE 62 (CONT'D)

CPU BOARD CONNECTOR PINS

CONN PIN	SIG IDENT	IC PIN	FUNCTION
J10-1A	IND	U13 - 6	OUTPUT
J10-1B	GND		
J10-2A	EXT1	U72 - 4	INPUT
J10-2B	LINK	U13 - 8	OUTPUT
J10-3A	EXT2	U72 - 3	INPUT
J10-3B	IND*	U13 - 5	INPUT
J10-4A	EXT3	U72 - 2	INPUT
J10-4B	MAR03*	U43 - 14	OUTPUT
J10-5A	PFEIN	U06 - 8	OUTPUT
J10-5B	MAR02*	U43 - 16	OUTPUT
J10-6A	HLTP*	U65 - 12	OUTPUT
J10-6B	MAR01*	U43 - 18	OUTPUT
J10-7A	EX*	U19 - 11	OUTPUT
J10-7B	MAR00*	U43 - 20	OUTPUT
J10-8A	FLAG1	U71 - 6	OUTPUT
J10-8B	QMAR*	U39 - 22	INPUT
J10-9A	DAT01	U37 - 17	I/O
J10-9B	ZERO	U09 - 7	OUTPUT
J10-10A	FLAG2	U71 - 8	OUTPUT
J10-10B	IR00	U25 - 11	OUTPUT
J10-11A	DAT04	U37 - 8	I/O
J10-11B	AI	U20 - 5	INPUT
J10-12A	DAT05	U37 - 7	I/O
J10-12B	IR01	U25 - 5	OUTPUT
J10-13A	DAT07	U37 - 3	I/O
J10-13B	MAR07*	U42 - 14	OUTPUT
J10-14A	I3	U19 - 8	OUTPUT
J10-14B	GND		

TABLE 63

J10-15A	ETIR*	U46 - 1	OUTPUT
J10-15B	MAR06*	U42 - 16	OUTPUT
J10-16A	I2	U56 - 4	OUTPUT
J10-16B	MAR05*	U42 - 18	OUTPUT
J10-17A	QMI*	U63 - 12	INPUT
J10-17B	MAR04*	U42 - 20	OUTPUT
J10-18A	CUND	U49 - 3	OUTPUT
J10-18B	TUR*	U20 - 8	OUTPUT
J10-19A	TEST*	U48 - 13	INPUT
J10-19B	TDR*	U20 - 11	OUTPUT
J10-20A	ELSB*	U70 - 4	OUTPUT
J10-20B	TIB*	U20 - 6	OUTPUT
J10-21A	ESPC*	U70 - 3	OUTPUT
J10-21B	DAT03	U37 - 13	I/O
J10-22A	ESTRT*	U70 - 2	OUTPUT
J10-22B	DAT02	U37 - 14	I/O
J10-23A	EBCH*	U70 - 1	OUTPUT
J10-23B	A	U13 - 1	INPUT
J10-24A	ARM	U63 - 1	INPUT
J10-24B	SAT*	U04 - 6	INPUT
J10-25A	HALT*	U48 - 9	OUTPUT
J10-25B	READY	U09 - 17	INPUT
J10-26A	EDR	U48 - 60	OUTPUT
J10-26B	TSSR	U09 - 18	INPUT
J10-27A	IAM*	U48 - 6	OUTPUT
J10-27B	DAT00	U37 - 18	I/O
J10-28A	GND		
J10-28B	GND		
J10-29A	IR*	U48 - 3	INPUT
J10-29B	EOUT*	U30 - 1	INPUT

J10-30A	ADV	U09 - 4	OUTPUT
J10-30B	A*	U38 - 15	INPUT
J10-31A	DAT12	U36 - 8	I/O
J10-31B	DAT11	U36 - 13	I/O
J10-32A	SVDC		
J10-32B	SVDC		
J10-33A	DAT13	U36 - 7	I/O
J10-33B	DAT09	U36 - 17	I/O
J10-34A	DAT10	U36 - 14	I/O
J10-34B	DAT08	U36 - 18	I/O
J10-35A	FOV	U06 - 6	OUTPUT
J10-35B	MAR11*	U40 - 14	OUTPUT
J10-36A	DAT14	U36 - 4	I/O
J10-36B	MAR10*	U40 - 16	OUTPUT
J10-37A	RPTOV*	U04 - 14	OUTPUT
J10-37B	GND		
J10-38A	DAT15	U36 - 3	I/O
J10-38B	MAR09*	U40 - 18	OUTPUT
J10-39A	DAT06	U37 - 4	I/O
J10-39B	MAR08*	U40 - 20	OUTPUT
J10-40A	I1	U64 - 11	OUTPUT
J10-40B	IR08	U25 - 12	OUTPUT
J10-41A	U29	U67 - 10	OUTPUT
J10-41B	SIGN	U09 - 8	OUTPUT
J10-42A	U28	U67 - 7	OUTPUT
J10-42B	COUT	U09 - 13	OUTPUT
J10-43A	U28*	U67 - 6	OUTPUT
J10-43B	IR02	U18 - 11	OUTPUT
J10-44A	BBUF		INPUT
J10-44B	U30*	U67 - 14	OUTPUT
J10-45A	GND		

TABLE 63 (CONT'D)

J10-45B	IR09	U25 - 4	OUTPUT
J10-46A	UMA9	U15 - 15	OUTPUT
J10-46B	GND		
J10-47A	UMA8	U15 - 19	OUTPUT
J10-47B	U27*	U67 - 3	OUTPUT
J10-48A	UMA7	U15 - 18	OUTPUT
J10-48B	U27	U67 - 2	OUTPUT
J10-49A	UMA6	U15 - 17	OUTPUT
J10-49B	U30	U67 - 15	OUTPUT
J10-50A	UMA5	U15 - 16	OUTPUT
J10-50B	POS	U60 - 1	INPUT
J10-51A	UMA4	U15 - 5	OUTPUT
J10-51B	IR03	U18 - 5	OUTPUT
J10-52A	UMA3	U15 - 4	OUTPUT
J10-52B	MAR15*	U39 - 14	OUTPUT
J10-53A	UMA2	U15 - 3	OUTPUT
J10-53B	MAR14*	U39 - 16	OUTPUT
J10-54A	UMA1	U15 - 2	OUTPUT
J10-54B	MAR13*	U39 - 18	OUTPUT
J10-55A	UMA0	U15 - 1	OUTPUT
J10-55B	GND		
J10-56A	VCC		
J10-56B	MAR12*	U39 - 20	OUTPUT

TABLE 63 (CONT'D)

APPENDIX F

TIMING AND CONTROL CARD DESCRIPTION

This Appendix contains:

- o Table of chip labels vs chip type,
- o Table of connections between chips and card connectors,
- o BLISS program for interchip connections, and
- o Description of each external connection for J9 timing and control card.

TIMING & CONTROL CHIP LABELS *Line 2866 of B-101*

CHIP ID		CHIP TYPE	
U4	=	74-LS-11	REQUIRED
U5	=	74-LS-08	REQUIRED
U13	=	74-S-10	REQUIRED
U21	=	74-S-37	REQUIRED
U32	=	74-LS-00	REQUIRED
U38	=	74-40	REQUIRED
U44	=	74-S-20	REQUIRED

TABLE 64

TIMING AND CONTROL CARD CHIP INTERCONNECTIONS

OUTPUT	DESTINATION	"SIG NAME"
P09 - 21B ->	U32 - 9, U38 - 13, U13 - 9, U44 - 4	"U27*"
P09 - 23C ->	U32 - 1, U32 - 5	"U27"
P09 - 24C ->	U32 - 2, U13 - 13	"U28*"
P09 - 21C ->	U32 - 4, U38 - 10, U32 - 10, U44 - 1	"U28"
P09 - 25C ->	U13 - 2	"U30*"
P09 - 4C ->	U13 - 11	"U30"
P09 - 18C ->	U13 - 1, U13 - 10, U44 - 2	"U29"
P09 - 15C ->	U44 - 5	"IND*"
U32 - 8 ->	P09 - 55A	"MEM*"
U13 - 12 ->	U04 - 3	
U32 - 3 ->	U04 - 5, P09 - 19C	"QIO*"
U13 - 8 ->	U05 - 5	
U32 - 6 ->	U05 - 4, P09 - 22C	"QII*"
U05 - 6 ->	P09 - 27B	"QI0IN*"
U44 - 6 ->	U38 - 9, U04 - 4, P09 - 27C	"MM*"
U04 - 6 ->	P09 - 30C	"EOUT*"
VCC ->	U38 - 12	
U38 - 8 ->	P09 - 25A	"QMI*"

TABLE 65

```

MODULE P2T (LANGUAGE(%PLISS36(BLISS36)
    %BLISS32(BLISS32)
    %BLISS16(BLISS16)),
    addressing_mode(external = long_relative,
        nonexternal = long_relative) ) =
BEGIN
GLOBAL ROUTINE PARTI2 : novalue =
    BEGIN %( FETCH PARTITION )%
%(
    THIS PARTITION PERFORMS THE FETCH PORTION OF AN
    INSTRUCTION CYCLE OF THE BDX 930 COMPUTER.
    IT CONTAINS THE FOLLOWING PARTS OF
    THE COMPUTER:
        FLAG AND STATUS REGISTER
        MEMORY DATA BUS
)%
    require "GLOBAL.r32";
    require "RTINEST.r32";

LOCAL
    A, B, C, D, E, YA, YB, QION, QIOINN, GND, J, K, KK;
LOCAL
    TT00, TT01, TT02, TT03, TT04, TT05, TT06, TT07, TT08, TT09,
    TT10, TT11, TT12, TT13, TT14, TT15;

external routine TC153,TC113,TC174,TC2451,TC2450,RWMEM,ALU,ALUT,TC257;
EXTERNAL JPART;

K = 571; J = 0;

```

```

!           TIMING AND CONTROL CARD CHIP INTERCONNECTIONS
!           PARTITION 2

!
!           OUTPUT           DESTINATION           "SIG NAME"
!
!
!           ! TIMING SIGNALS FOR MEMORY FETCH AND STORE
!
!           P09 - 21R ->  U32 - 9, U38 - 13, U13 - 9,           "U27*"
!                           U44 - 4
!           P09 - 23C ->  U32 - 1, U32 - 5                     "U27"
!           P09 - 24C ->  U32 - 2, U13 - 13                     "U28*"
!           P09 - 21C ->  U32 - 4, U38 - 10, U32 - 10,         "U28"
!                           U44 - 1
!           P09 - 25C ->  U13 - 2                                 "U30*"
!           P09 - 4C ->   U13 - 11                               "U30"
!           P09 - 18C ->  U13 - 1, U13 - 10, U44 - 2           "U29"
!           P09 - 15C ->  U44 - 5                               "IND*"

```



```

NAND(MEMN,U28,U27N); NAND3(A,U28N,U29,U30N);

!      U32 = 8  ->  P09 = 55A      "MEM*"
!      U13 = 12 ->  U04 = 3

NAND(QION,U27,U28N); NAND3(QI0INN,U27N,U29,U30);

!      U32 = 3  ->  U04 = 5, P09 = 19C      "QI0*"
!      U13 = 8  ->  U05 = 5
!      U32 = 6  ->  U05 = 4, P09 = 22C      "QI1*"
!      U05 = 6  ->  P09 = 27B      "QI0IN*"

NAND4(MMN,U27N,U28,U29,INDN);

!      U44 = 6  ->  U38 = 9, U04 = 4, P09 = 27C      "MM*"

EOUTN = AND3(QION,A,MMN); INV(MM,MMN);

!      U04 = 6  ->  P09 = 30C      "EOUT*"

NAND3(QMIN,U27N,U28,MMN);

!      VCC      ->  U38 = 12
!      U38 = 8  ->  P09 = 25A      "QMI*"

END;      ! ROUTINE PARTN2
END
ELUDOM      ! P2

```

TABLE 66 (CONT'D)

TIMING AND CONTROL BOARD CONNECTOR PINS

CONN PIN	SIG IDENT	IC PIN	FUNCTION
J9-4C	U30	U13 - 11	INPUT
J9-15C	IND*	U44 - 5	INPUT
J9-18C	U29	U13 - 1	INPUT
J9-19C	Q10*	U04 - 5	OUTPUT
J9-21B	U27*	U32 - 9	INPUT
J9-21C	U28	U32 - 4	INPUT
J9-22C	Q11*	U32 - 6	OUTPUT
J9-23C	U27	U32 - 1	INPUT
J9-24C	U28*	U32 - 2	INPUT
J9-25A	QMI*	U38 - 8	OUTPUT
J9-25C	U30*	U13 - 2	INPUT
J9-27B	Q10IN*	U05 - 6	OUTPUT
J9-27C	MM*	U38 - 9	OUTPUT
J9-30C	EOUT*	U04 - 6	OUTPUT
J9-55A	MEM*	U32 - 8	OUTPUT

TABLE 67

APPENDIX G

PROM DATA

The BDX930 contains three sets of PROMS whose functions are:

- o Sequencer Control (54S288)
- o Microcode Starting Address (54S472), and
- o Microcode Memory (Seven 54S472) -

The data to be entered in these PROMS is included in this Appendix.

LOCATION					CONTENTS									FUNCTION
U40	U41	U42	CONDITION	HALT+INT	PIN #	EBCH*	ESTRT*	EEPC*	ELSB*	EMSB*	SB	SA	ETIR*	
0	0	0	0	0		1	1	0	0	0	1	1	0	POWER ON
			0	1		1	1	0	0	0	1	1	0	
			1	0		1	1	0	0	0	1	1	0	
			1	1		1	1	0	0	0	1	1	0	
0	0	1	0	0		1	0	1	1	0	1	0	0	EXTENDED INSTR.
			0	1		1	0	1	1	0	1	0	0	
			1	0		1	0	1	1	0	1	0	0	
			1	1		1	0	1	1	0	1	0	0	
0	1	0	0	0		1	0	1	1	0	1	1	0	RESTART
			0	1		1	1	0	0	0	1	1	1	
			1	0		1	0	1	1	0	1	1	0	
			1	1		1	1	0	0	0	1	1	1	
0	1	1	0	0		0	1	1	0	1	0	0	1	CONDITIONAL START
			0	1		0	1	1	0	1	0	0	1	
			1	0		1	0	1	1	0	1	1	0	
			1	1		1	1	0	0	0	1	1	1	
1	0	0	0	0		0	1	1	0	1	0	1	1	CONDITIONAL BRANCH
			0	1		0	1	1	0	1	0	1	1	
			1	0		0	1	1	0	1	0	1	1	
			1	1		0	1	1	0	1	0	1	1	
1	0	1	0	0		0	0	0	0	0	0	0	0	SPARE
			0	1		0	0	0	0	0	0	0	0	
			1	0		0	0	0	0	0	0	0	0	
			1	1		0	0	0	0	0	0	0	0	
1	1	0	0	0		1	0	1	1	0	1	0	0	EXECUTE REGISTER
			0	1		1	0	1	1	0	1	0	0	
			1	0		1	0	1	1	0	1	0	0	
			1	1		1	0	1	1	0	1	0	0	
1	1	1	0	0		0	1	1	0	1	1	0	1	CONDITIONAL MULTIWAY
			0	1		0	1	1	0	1	1	0	1	
			1	0		0	1	1	0	1	1	0	1	
			1	1		0	1	1	0	1	1	0	1	

SEQUENCER CONTROL PROM

TABLE 68

				ADDRESS (DECIMAL)	
				CONTENTS (DECIMAL)	
				ADDRESS (BINARY)	
				CONTENTS (BINARY)	
0	33	0 000 00 00	00100001	TRA	
1	14	0 000 00 01	00001110	DECEQ	
2	37	0 000 00 10	00100101	LCM	
3	12	0 000 00 11	00001100	RIS	
4	25	0 000 01 00	00010111	CONT	
5	15	0 000 01 01	00001111	DECNE	
6	35	0 000 01 10	00100011	AND	
7	13	0 000 01 11	00001101	RLL	
8	40	0 000 10 00	00101000	ADDR	
9	32	0 000 10 01	00100000	IR	
10	36	0 000 10 10	00100100	OR	
11	28	0 000 10 11	00011100	MPY	
12	41	0 000 11 00	00101001	SUBR	
13	38	0 000 11 01	00100110	ACM	
14	34	0 000 11 10	00100010	CMPR	
15	29	0 000 11 11	00011101	DIV	
16	173	0 001 00 00	10110010	JU	
17	186	0 001 00 01	10111010	JSA0	
18	190	0 001 00 10	10111110	JSA1	
19	66	0 001 00 11	01000010	JMA0	
20	180	0 001 01 00	10110100	JU	
21	187	0 001 01 01	10111011	JSA0	
22	191	0 001 01 10	10111111	JSA1	
23	67	0 001 01 11	01000011	JMAC	
24	182	0 001 10 00	10110110	JU	
25	189	0 001 10 01	10111101	JSA0	
26	192	0 001 10 10	11000000	JSA1	
27	68	0 001 10 11	01000100	JMA0	
28	184	0 001 11 00	10111000	JU	
29	188	0 001 11 01	10111100	JSA0	
30	193	0 001 11 10	11000001	JSA1	
31	69	0 001 11 11	01000101	JMAC	
32	74	0 010 00 00	01001010	ADD	
33	74	0 010 00 01	01001010	ADD	
34	74	0 010 00 10	01001010	ADD	
35	74	0 010 00 11	01001010	ADD	
36	75	0 010 01 00	01001011	ADD	
37	75	0 010 01 01	01001011	ADD	
38	75	0 010 01 10	01001011	ADD	
39	75	0 010 01 11	01001011	ADD	
40	76	0 010 10 00	01001100	ADD	
41	76	0 010 10 01	01001100	ADD	
42	76	0 010 10 10	01001100	ADD	
43	76	0 010 10 11	01001100	ADD	
44	77	0 010 11 00	01001101	ADD	
45	77	0 010 11 01	01001101	ADD	
46	77	0 010 11 10	01001101	ADD	
47	77	0 010 11 11	01001101	ADD	
48	78	0 011 00 00	01001110	SUB	
49	78	0 011 00 01	01001110	SUB	
50	78	0 011 00 10	01001110	SUB	
51	78	0 011 00 11	01001110	SUB	
52	78	0 011 01 00	01001111	SUB	
53	79	0 011 01 01	01001111	SUB	
54	79	0 011 01 10	01001111	SUB	

TABLE 69

55	79	0	011	01	11	01001111	SUB
56	80	0	011	10	00	01010000	SUB
57	80	0	011	10	01	01010000	SUB
58	80	0	011	10	10	01010000	SUB
59	80	0	011	10	11	01010000	SUB
60	81	0	011	11	00	01010001	SUB
61	81	0	011	11	01	01010001	SUB
62	81	0	011	11	10	01010001	SUB
63	81	0	011	11	11	01010001	SUB
64	70	0	100	00	00	01000110	CMP
65	70	0	100	00	01	01000110	CMP
66	70	0	100	00	10	01000110	CMP
67	70	0	100	00	11	01000110	CMP
68	71	0	100	01	00	01000111	CMP
69	71	0	100	01	01	01000111	CMP
70	71	0	100	01	10	01000111	CMP
71	71	0	100	01	11	01000111	CMP
72	72	0	100	10	00	01001000	CMP
73	72	0	100	10	01	01001000	CMP
74	72	0	100	10	10	01001000	CMP
75	72	0	100	10	11	01001000	CMP
76	73	0	100	11	00	01001001	CMP
77	73	0	100	11	01	01001001	CMP
78	73	0	100	11	10	01001001	CMP
79	73	0	100	11	11	01001001	CMP
80	42	0	101	00	00	00101010	LOAD
81	42	0	101	00	01	00101010	LOAD
82	42	0	101	00	10	00101010	LOAD
83	42	0	101	00	11	00101010	LOAD
84	43	0	101	01	00	00101011	LOAD
85	43	0	101	01	01	00101011	LOAD
86	43	0	101	01	10	00101011	LOAD
87	43	0	101	01	11	00101011	LOAD
88	44	0	101	10	00	00101100	LOAD
89	44	0	101	10	01	00101100	LOAD
90	44	0	101	10	10	00101100	LOAD
91	44	0	101	10	11	00101100	LOAD
92	45	0	101	11	00	00101101	LOAD
93	45	0	101	11	01	00101101	LOAD
94	45	0	101	11	10	00101101	LOAD
95	45	0	101	11	11	00101101	LOAD
96	46	0	110	00	00	00101110	STO
97	46	0	110	00	01	00101110	STO
98	46	0	110	00	10	00101110	STO
99	46	0	110	00	11	00101110	STO
100	47	0	110	01	00	00101111	STO
101	47	0	110	01	01	00101111	STO
102	47	0	110	01	10	00101111	STO
103	47	0	110	01	11	00101111	STO
104	48	0	110	10	00	00110000	STO
105	48	0	110	10	01	00110000	STO
106	48	0	110	10	10	00110000	STO
107	48	0	110	10	11	00110000	STO
108	49	0	110	11	00	00110001	STO
109	49	0	110	11	01	00110001	STO
110	49	0	110	11	10	00110001	STO
111	49	0	110	11	11	00110001	STO
112	27	0	111	00	00	00011011	OC
113	27	0	111	00	01	00011011	OC
114	27	0	111	00	10	00011011	OC
115	27	0	111	00	11	00011011	OC
116	26	0	111	01	00	00011010	ID
117	26	0	111	01	01	00011010	ID

TABLE 69 (CONT'D)

118	26	0	111	01	10	00011010	10
119	26	0	111	01	11	00011010	10
120	25	0	111	10	00	00011001	00
121	25	0	111	10	01	00011001	00
122	25	0	111	10	10	00011001	00
123	25	0	111	10	11	00011001	00
124	24	0	111	11	00	00011000	1SW
125	87	0	111	11	01	01010111	LUM
126	87	0	111	11	10	01010111	STM
127	100	0	111	11	11	01100100	EXTEND (STACKING)
128	9	1	000	00	00	00001001	SLSA
129	8	1	000	00	01	00001000	SLLA
130	18	1	000	00	10	00010010	SKGT
131	16	1	000	00	11	00010000	SKLT
132	10	1	000	01	00	00001010	SLSL
133	11	1	000	01	01	00001011	SLLL
134	20	1	000	01	10	00010100	SKGE
135	19	1	000	01	11	00010011	SKLE
136	4	1	000	10	00	00000100	SKSA
137	5	1	000	10	01	00000101	SKLA
138	21	1	000	10	10	00010101	SKEQ
139	22	1	000	10	11	00010110	SKID
140	7	1	000	11	00	00000111	SRSL
141	6	1	000	11	01	00000110	SRLL
142	17	1	000	11	10	00010001	SKNE
143	39	1	000	11	11	00100111	IAR
144	179	1	001	00	00	10110011	JU*
145	186	1	001	00	01	10111010	JSA0*
146	190	1	001	00	10	10111110	JSA1*
147	66	1	001	00	11	01000010	JMA0*
148	181	1	001	01	00	10110101	JU*
149	187	1	001	01	01	10111011	JSA0*
150	191	1	001	01	10	10111111	JSA1*
151	67	1	001	01	11	01000011	JMA0*
152	183	1	001	10	00	10110111	JU*
153	125	1	001	10	01	01111101	JSA0*
154	192	1	001	10	10	11000000	JSA1*
155	68	1	001	10	11	01000100	JMA0*
156	185	1	001	11	00	10111001	JU*
157	188	1	001	11	01	10111100	JSA0*
158	105	1	001	11	10	01101001	JSA1*
159	69	1	001	11	11	01000101	JMA0*
160	74	1	010	00	00	01001010	ADD*
161	74	1	010	00	01	01001010	ADD*
162	74	1	010	00	10	01001010	ADD*
163	74	1	010	00	11	01001010	ADD*
164	75	1	010	01	00	01001011	ADD*
165	75	1	010	01	01	01001011	ADD*
166	75	1	010	01	10	01001011	ADD*
167	75	1	010	01	11	01001011	ADD*
168	76	1	010	10	00	01001100	ADD*
169	76	1	010	10	01	01001100	ADD*
170	76	1	010	10	10	01001100	ADD*
171	76	1	010	10	11	01001100	ADD*
172	77	1	010	11	00	01001101	ADD*
173	77	1	010	11	01	01001101	ADD*
174	77	1	010	11	10	01001101	ADD*
175	77	1	010	11	11	01001101	ADD*
176	78	1	011	00	00	01001110	SUB*
177	78	1	011	00	01	01001110	SUB*
178	78	1	011	00	10	01001110	SUB*
179	78	1	011	00	11	01001110	SUB*
180	79	1	011	01	00	01001111	SUB*

TABLE 69 (CONT'D)

181	72	1	011	01	00	01001111	SUB*
182	79	1	011	01	10	01001111	SUB*
183	79	1	011	01	11	01001111	SUB*
184	80	1	011	10	00	01010000	SUB*
185	80	1	011	10	01	01010000	SUB*
186	80	1	011	10	10	01010000	SUB*
187	80	1	011	10	11	01010000	SUB*
188	81	1	011	11	00	01010001	SUB*
189	81	1	011	11	01	01010001	SUB*
190	81	1	011	11	10	01010001	SUB*
191	81	1	011	11	11	01010001	SUB*
192	70	1	100	00	00	01000110	CMP*
193	70	1	100	00	01	01000110	CMP*
194	70	1	100	00	10	01000110	CMP*
195	70	1	100	00	11	01000110	CMP*
196	71	1	100	01	00	01000111	CMP*
197	71	1	100	01	01	01000111	CMP*
198	71	1	100	01	10	01000111	CMP*
199	71	1	100	01	11	01000111	CMP*
200	72	1	100	10	00	01001000	CMP*
201	72	1	100	10	01	01001000	CMP*
202	72	1	100	10	10	01001000	CMP*
203	72	1	100	10	11	01001000	CMP*
204	73	1	100	11	00	01001001	CMP*
205	73	1	100	11	01	01001001	CMP*
206	73	1	100	11	10	01001001	CMP*
207	73	1	100	11	11	01001001	CMP*
208	42	1	101	00	00	00101010	LOAD*
209	42	1	101	00	01	00101010	LOAD*
210	42	1	101	00	10	00101010	LOAD*
211	42	1	101	00	11	00101010	LOAD*
212	43	1	101	01	00	00101011	LOAD*
213	43	1	101	01	01	00101011	LOAD*
214	43	1	101	01	10	00101011	LOAD*
215	43	1	101	01	11	00101011	LOAD*
216	44	1	101	10	00	00101100	LOAD*
217	44	1	101	10	01	00101100	LOAD*
218	44	1	101	10	10	00101100	LOAD*
219	44	1	101	10	11	00101100	LOAD*
220	45	1	101	11	00	00101101	LOAD*
221	45	1	101	11	01	00101101	LOAD*
222	45	1	101	11	10	00101101	LOAD*
223	45	1	101	11	11	00101101	LOAD*
224	46	1	110	00	00	00101110	STO*
225	46	1	110	00	01	00101110	STO*
226	46	1	110	00	10	00101110	STO*
227	46	1	110	00	11	00101110	STO*
228	47	1	110	01	00	00101111	STO*
229	47	1	110	01	01	00101111	STO*
230	47	1	110	01	10	00101111	STO*
231	47	1	110	01	11	00101111	STO*
232	48	1	110	10	00	00110000	STO*
233	48	1	110	10	01	00110000	STO*
234	48	1	110	10	10	00110000	STO*
235	48	1	110	10	11	00110000	STO*
236	49	1	110	11	00	00110001	STO*
237	49	1	110	11	01	00110001	STO*
238	49	1	110	11	10	00110001	STO*
239	49	1	110	11	11	00110001	STO*
240	84	1	111	00	00	01010100	DADDR
241	83	1	111	00	01	01010011	DSUR
242	2	1	111	00	10	00000010	
243	30	1	111	00	11	00011110	LXOR

TABLE 69 (CONT'D)

244	26	1	111	01	00	00011010	ISR
245	26	1	111	01	01	00011010	ISR
246	26	1	111	01	10	00011010	ISR
247	26	1	111	01	11	00011010	ISR
248	25	1	111	10	00	00011001	OSR
249	25	1	111	10	01	00011001	OSR
250	25	1	111	10	10	00011001	OSR
251	25	1	111	10	11	00011001	OSR
252	218	1	111	11	00	11011010	HALT
253	106	1	111	11	01	01101010	RET
254	2	1	111	11	10	00000010	
255	86	1	111	11	11	01010110	MACRO
256	240	100000000				11110000	RSRT
257	152	100000001				10011000	
258	152	100000010				10011000	
259	152	100000011				10011000	
260	152	100000100				10011000	
261	152	100000101				10011000	
262	152	100000110				10011000	
263	152	100000111				10011000	
264	152	100001000				10011000	
265	152	100001001				10011000	
266	152	100001010				10011000	
267	152	100001011				10011000	
268	152	100001100				10011000	
269	152	100001101				10011000	
270	152	100001110				10011000	
271	152	100001111				10011000	
272	96	100010000				01100000	JSS
273	88	100010001				01011000	MULTI
274	95	100010010				01011111	RPS
275	152	100010011				10011000	
276	97	100010100				01100001	JSS
277	89	100010101				01011001	MULTI
278	152	100010110				10011000	
279	152	100010111				10011000	
280	98	100011000				01100010	JSS
281	90	100011001				01011010	MULTI
282	152	100011010				10011000	
283	152	100011011				10011000	
284	99	100011100				01100011	JSS
285	91	100011101				01011011	MULTI
286	156	100011110				10011100	EXECR
287	152	100011111				10011000	
288	85	100100000				01010101	DEPY
289	82	100100001				01010010	DACM
290	152	100100010				10011000	
291	152	100100011				10011000	
292	152	100100100				10011000	
293	152	100100101				10011000	
294	152	100100110				10011000	
295	152	100100111				10011000	
296	152	100101000				10011000	
297	152	100101001				10011000	
298	152	100101010				10011000	
299	152	100101011				10011000	
300	152	100101100				10011000	
301	152	100101101				10011000	
302	152	100101110				10011000	
303	152	100101111				10011000	
304	152	100110000				10011000	
305	152	100110001				10011000	
306	152	100110010				10011000	

TABLE 69 (CONT'D)

307	152	100110011	10011000	
308	152	100110100	10011000	
309	152	100110101	10011000	
310	152	100110110	10011000	
311	152	100110111	10011000	
312	152	100111000	10011000	
313	152	100111001	10011000	
314	152	100111010	10011000	
315	152	100111011	10011000	
316	152	100111100	10011000	
317	152	100111101	10011000	
318	152	100111110	10011000	
319	152	100111111	10011000	
320	107	101000000	01101011	PADDB
321	101	101000001	01100101	POPM
322	102	101000010	01100110	PUSHB
323	152	101000011	10011000	
324	152	101000100	10011000	
325	103	101000101	01100111	POPF
326	104	101000110	01101000	PUSHF
327	152	101000111	10011000	
328	152	101001000	10011000	
329	152	101001001	10011000	
330	152	101001010	10011000	
331	152	101001011	10011000	
332	152	101001100	10011000	
333	152	101001101	10011000	
334	152	101001110	10011000	
335	152	101001111	10011000	
336	152	101010000	10011000	
337	152	101010001	10011000	
338	152	101010010	10011000	
339	152	101010011	10011000	
340	152	101010100	10011000	
341	152	101010101	10011000	
342	152	101010110	10011000	
343	152	101010111	10011000	
344	152	101011000	10011000	
345	152	101011001	10011000	
346	152	101011010	10011000	
347	152	101011011	10011000	
348	152	101011100	10011000	
349	152	101011101	10011000	
350	152	101011110	10011000	
351	152	101011111	10011000	
352	152	101100000	10011000	
353	152	101100001	10011000	
354	152	101100010	10011000	
355	152	101100011	10011000	
356	152	101100100	10011000	
357	152	101100101	10011000	
358	152	101100110	10011000	
359	152	101100111	10011000	
360	152	101101000	10011000	
361	152	101101001	10011000	
362	152	101101010	10011000	
363	152	101101011	10011000	
364	152	101101100	10011000	
365	152	101101101	10011000	
366	152	101101110	10011000	
367	152	101101111	10011000	
368	152	101110000	10011000	
369	152	101110001	10011000	

TABLE 69 (CONT'D)

370	152	101110010	10011000	
371	152	101110011	10011000	
372	152	101110100	10011000	
373	152	101110101	10011000	
374	152	101110110	10011000	
375	152	101110111	10011000	
376	152	101111000	10011000	
377	152	101111001	10011000	
378	152	101111010	10011000	
379	152	101111011	10011000	
380	152	101111100	10011000	
381	93	101111101	01011101	(LDM)
382	94	101111110	01011110	(STM)
383	152	101111111	10011000	
384	242	110000000	11111000	TEST
385	241	110000001	11110001	TEST
386	244	110000010	11110100	TEST
387	217	110000011	11011001	TEST
388	250	110000100	11111010	TEST
389	249	110000101	11111001	TEST
390	245	110000110	11110101	TEST
391	216	110000111	11011000	TEST
392	251	110001000	11111011	TEST
393	242	110001001	11110010	TEST
394	246	110001010	11110110	TEST
395	215	110001011	11010111	TEST
396	252	110001100	11111100	TEST
397	243	110001101	11110011	TEST
398	247	110001110	11110111	TEST
399	214	110001111	11010110	TEST
400	96	110010000	01100000	JSS
401	92	110010001	01011100	MULTI
402	152	110010010	10011000	
403	152	110010011	10011000	
404	97	110010100	01100001	JSS
405	89	110010101	01011001	MULTI
406	152	110010110	10011000	
407	152	110010111	10011000	
408	98	110011000	01100010	JSS
409	90	110011001	01011010	MULTI
410	152	110011010	10011000	
411	152	110011011	10011000	
412	99	110011100	01100011	JSS
413	91	110011101	01011011	MULTI
414	152	110011110	10011000	
415	152	110011111	10011000	
416	152	110100000	10011000	
417	152	110100001	10011000	
418	152	110100010	10011000	
419	152	110100011	10011000	
420	152	110100100	10011000	
421	152	110100101	10011000	
422	152	110100110	10011000	
423	152	110100111	10011000	
424	152	110101000	10011000	
425	152	110101001	10011000	
426	152	110101010	10011000	
427	152	110101011	10011000	
428	152	110101100	10011000	
429	152	110101101	10011000	
430	152	110101110	10011000	
431	152	110101111	10011000	
432	152	110110000	10011000	

TABLE 69 (CONT'D)

434	152	110110010	10011000
435	152	110110011	10011000
436	152	110110100	10011000
437	152	110110101	10011000
438	152	110110110	10011000
439	152	110110111	10011000
440	152	110111000	10011000
441	152	110111001	10011000
442	152	110111010	10011000
443	152	110111011	10011000
444	152	110111100	10011000
445	152	110111101	10011000
446	152	110111110	10011000
447	152	110111111	10011000
448	152	111000000	10011000
449	152	111000001	10011000
450	152	111000010	10011000
451	152	111000011	10011000
452	152	111000100	10011000
453	152	111000101	10011000
454	152	111000110	10011000
455	152	111000111	10011000
456	152	111001000	10011000
457	152	111001001	10011000
458	152	111001010	10011000
459	152	111001011	10011000
460	152	111001100	10011000
461	152	111001101	10011000
462	152	111001110	10011000
463	152	111001111	10011000
464	152	111010000	10011000
465	152	111010001	10011000
466	152	111010010	10011000
467	152	111010011	10011000
468	152	111010100	10011000
469	152	111010101	10011000
470	152	111010110	10011000
471	152	111010111	10011000
472	152	111011000	10011000
473	152	111011001	10011000
474	152	111011010	10011000
475	152	111011011	10011000
476	152	111011100	10011000
477	152	111011101	10011000
478	152	111011110	10011000
479	152	111011111	10011000
480	152	111100000	10011000
481	152	111100001	10011000
482	152	111100010	10011000
483	152	111100011	10011000
484	152	111100100	10011000
485	152	111100101	10011000
486	152	111100110	10011000
487	152	111100111	10011000
488	152	111101000	10011000
489	152	111101001	10011000
490	152	111101010	10011000
491	152	111101011	10011000
492	152	111101100	10011000
493	152	111101101	10011000
494	152	111101110	10011000
495	152	111101111	10011000

TABLE 69 (CONT'D)

497	152	111110000	10011000
498	152	111110001	10011000
499	152	111110010	10011000
500	152	111110011	10011000
501	152	111110100	10011000
502	152	111110101	10011000
503	152	111110110	10011000
504	152	111110111	10011000
505	152	111111000	10011000
506	152	111111001	10011000
507	152	111111010	10011000
508	152	111111011	10011000
509	152	111111100	10011000
510	152	111111101	10011000
511	216	111111110	10011000
		111111111	11011010 HALT

TABLE 69 (CONT'D)

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

APPENDIX H
CONTENTS OF THE COMPUTER TAPE SUPPLIED
FOR SOURCE CODE DISTRIBUTION

In order to facilitate the transfer of data included in this report, a computer tape was prepared with all pertinent emulation descriptions. The tape was created on a VAX 11/780 and is 1600 BPI.

The tape contains the following data sets:

25LS377.QM1;1	54125.QM1;1	54LS00.QM1;1	54LS02.QM1;1
54LS08.QM1;1	54LS113.QM1;1	54LS151.QM1;2	54LS153.QM1;1
54LS158.QM1;1	54LS169.QM1;2	54LS175.QM1;1	54LS245.QM1;1
54LS253.QM1;1	54LS273.QM1;1	54LS352.QM1;1	54LS367.QM1;1
54LS374.QM1;1	54LS472.QM1;1	54LS86.QM1;1	54S00.QM1;1
54S02.QM1;1	54S04.QM1;1	54S151.QM1;1	54S288.QM1;1
54S32.QM1;1	54S472.QM1;1	7440.QM1;1	74LS00.QM1;1
74LS08.QM1;1	74LS11.QM1;1	74S10.QM1;1	74S20.QM1;1
74S37.QM1;1	9407.QM1;1	AM2901A.QM1;1	AM2902.QM1;1
ASGNTAR.QM1;1	CPUCARD.QM1;4	CPUCDN.QM1;2	CPULAB.QM1;1
INTRCON.QM1;3	INTRTAB.QM1;3	TCCARD.QM1;5	TCCON.QM1;1
TCLAR.QM1;1			

In addition, the tape contains PROM data in octal contained in the following sets:

MICRO. MEM,

SEQUENCE. MEM, and

STARTADDR. MEM

listed in sequence by address.

Data set SELF.TST contains the self-test supplied in octal form. Please note that:

- o Each line starts with a memory address followed by its contents and the contents of succeeding locations, separated by commas.
- o The read only portion of the program has been relocated from the memory locations given in Appendix I to memory locations between 1000 and 2000 (octal). This was done for compactness of memory in the emulator.

- o RAM locations have been relocated between 2000 and 2200 (octal).
The address of CRSLT (the self-test result word) is 2120 octal.
- o Execution of the self-test program begins at location 0 which contains a jump indirect instruction (to location 1041) and ends at location 1440 octal which contains a halt instruction.

The above changes were made to the self-test program to facilitate its use in the emulator without a calling program.

APPENDIX I
SELF-TEST PROGRAM

CPU SELF TEST

This CPU test is directed toward the individual blocks of hardware that comprise the CPU. The strategy is to break the 53 bit microprogram instruction down to its functional fields and set up tests to cover them individually. These fields either control arithmetic unit functions or switch the multiplexers that control data flow. Test computations are set up using instructions that toggle each bit of these fields. The result of the computation is then tested by performing a cyclic check sum of the contents of all of the registers. The 16 accumulators are mechanized by means of the 16 x 4 dual port scratch pads in each of the 4 2901 bit slice processor devices. Therefore, the accumulators and all of the interaccumulator data transfers can be tested more efficiently as a dual port RAM device. The pipeline register is tested by generating enough codes to toggle each bit. This is relatively easy and falls out from the other tests. The test of the microprogram start address PROM also is a fallout from the other tests. The remainder of the CPU consists primarily of the microprogram memory, which is the most difficult part of test. Here, the coverage of earlier tests is analyzed and a set of instructions previously unused are combined in a test computation, followed by a cyclic sum check of the contents of all the accumulators. This set of instructions is adjusted to bring the coverage up to 95% for the overall CPU.

The test result (pass or fail) is transmitted to the driver program units in RAM words TSSTC and CRSLT which are elements of segment TSCPU contained in TST (Test Status Table). These words are initialized at zero and incremented as each test step is passed. This test is composed of the individual tests listed below. They are not necessarily run in sequence because it was found necessary to run some of the steps of different tests concurrently.

<u>TEST</u>	<u>DEVICE TESTED</u>
1) Accumulator Scratch Pad Test	scratch pad of 2901
2) Memory Address Processor Test	9407 Address processor
3) ALU Test	ALU of 2901
4) Microprogram Memory Test	PROMS containing microprogram

Successful execution of this self-test program is indicated by a return to its calling program with a value of ten stored in memory location CRSLT. This program requires 346 words of memory and slightly over 2050 microcycles of the BDX930 for its execution on a fault-free processor.

1) ACCUMULATOR SCRATCH PAD TEST

The 16 accumulators are mechanized by means of the 16 x 4 dual port scratch pads in each of the 4 2901 bit slice processor devices. Therefore, a cyclic check sum is an effective method of testing the state at the contents of the accumulators. That is, the contents of the accumulators is checked by summing the contents of all the accumulators (except 15) and comparing the result to the correct value contained in a table. A15 is the stack pointer and requires a special test. The addition is performed in the unsaturated mode. To enable this test to detect if the contents of the registers are interchanged the result of each addition is rotated. This cyclic sumcheck is also used to test the state of the accumulators after each test calculation. This detects unexpected results such as nonspecified registers being affected. The scratch pad has 2 ports A and B. The A port is used for output only while the B port is used for both input and output. The infrequently used B output port requires additional testing to detect failures in the connections between the memory cells and the B output port multiplexer. The test proceeds as follows:

- 1) Using the LOAD instruction, place the starting address of a table containing 16 test words, into A0. These test words form a pattern of 16 words that exercise each bit of the data paths to and from the scratch pad.
- 2) Using the LOAD instruction with the relative to A0 address mode load the third word in the table into A1. Then increment A0.
- 3) Using the TRA instruction transfer the contents of A1 to A2.
- 4) Repeat (2) and (3) for A3 through A15. Accumulators A2 through A15 now contain the first 14 test words listed in the table. A15 is loaded with the address of the stack.
- 5) Test A15 (the stack pointer) by complementing its contents using the LCM instruction, adding the result to A1 (which contains the original contents), and then incrementing the result in A1 by 1. The result is then tested for 0.
- 6) Use the LOAD instruction in the relative to P mode to load the first two test words in the table to A0 and A1.
- 7) Rotate A0 left 1 bit using RLS
- 8) Add A0 to A1 using ADDR in the unsaturated mode. The first instruction of this test is a CONT 1S. This sets F1 and engages the unsaturated mode.
- 9) Repeat (7) and (8) for A2 through A14.
- (10) Compare the result to the 17th value of the table using CMP instruction in the relative to P mode, and if they agree increment CRSLT. At this point the data input path, A data output port, A address multiplexer, B address multiplexer, and the data path from A output to data path from A output to data input have been tested. However, the scratch pad memory cells must be tested for stuck at failures by the next sequence of test steps.

- (11) Restore the value for A0 from the table.
- (12) Using LCM complement the contents of all the accumulators except A15.
- (13) Repeat (7), (8) and (9) by calling these steps as a procedure by means of the JSS instruction. The return is accomplished by means of the RPS instruction. The result is then compared to the 18th value of the table. If they agree, increment CRSLT. A15 is retained as the stack pointer. Hence, is unaffected by this procedure.
- (14) The data flow through the B output port is untested by the above test procedure. Therefore, the contents of A0 and A15 are exchanged using two IR instructions. The contents of these registers are complemented and exchanged again using the IR instruction. This exercises each bit of both the data and the address paths. This is inserted in the ALU test and the result is tested as the overall ALU test result.
- (15) The connections between the memory cells and the B multiplexer is tested by means of the CLA instruction. This instruction is really an IR, specifying the same register in the two fields. The micro code for this instruction actually performs a sum and difference computation on the contents of the specified accumulators. The contents of the accumulator is set to zero only if the same value is received from both A and B multiplexers. A PUSHM and a PADDM are used to sum the contents of the accumulators (A1 through A14). The result is then tested for 0.

As indicated earlier the rotation of (7) is used to test the order of the test values as they appear in the accumulators. It detects if the values of two accumulators are interchanged due to a failure of the A or B multiplexers. This procedure is not only a thorough test of the scratch pad RAM, it also tests the ALU Data Source Selector, Q Register, Q Shift in the left direction, the F bus, and the + and OR ALU functions, Tables 71 through 75 indicate the test coverage of the microinstruction fields by the instructions used in this

test. Table Z6 lists the instructions used in the order that they are first used. These tables give a good indication of the effectiveness of this test.

2) MEMORY ADDRESS PROCESSOR TEST

The memory address is computed in the 4 9407 bit slice processor devices. The Program Counter (P) is one of 4 registers contained in this device. The accumulator Scratch pad test has already tested a portion of these devices. However, it has not tested the most significant bits because it has been addressing either words inside the memory segments TSCPU or CPUT. The most significant bits can only be exercised by specifying another unit of CPUTM in a segment of memory whose address is the logical component (for the most significant 8 bits) of CPUT. The P,T,MAR registers, full adder, and the 3 input MUX are tested as follows:

- 1) Use LOAD to load A0 with (start of test segment -XX), XX is set at -1 to exercise the D0 to D3 inputs to the full adder. They are normally zero.
- 2) Use LOAD 1,XX+4.0 to load a test word contained in segment of memory whose address is the logical complement of the most significant 8 bits (7 through 15), of the address of CPUT. This exercises the most significant bits of the T register.
- 3) Compare this result to a test word and if they agree increment CRSLT.
- 4) Reset the T register by loading an accumulator with a word with a low memory address (from within CPUT).
- 5) Load A1 with the starting address of the test unit -XX. Then execute a JSA1 XX.1 to call the unit. The test unit contains a JMA0 to the next instruction followed by a RET. The JSA1 exercises the P register but not the data path to the accumulator scratch pad. Hence, the JMA0 is required. The JSA1 is followed by a jump to the end of CPUT to provide a test of the jump. This jump is followed by the normal point

of return that increments CRSLT. CRSLT is also incremented in the test unit to show that it reached there.

The value of XX should be chosen to exercise all the bits of the binary adder of the 9407. Table II shown below, lists the coverage of the Memory Address Processor operations. Also listed are additional that are required to complete the coverage. They will be included in the ALU test.

COVERAGE OF THE MEMORY ADDRESS PROCESSOR OPERATIONS

FIELD	COVERAGE
U20,U21,U22,U23,U24	
0	CONT
1	JSA1
2	not testible
3	PUSHF 1
4	not used
5	RPS
6	not used
7	STO
8	not used
9	not used
A	JSA1
B	STO
C	SKEQ
D	SKEQ
E	not used
F	IAR
10	STO
11	CONT
12	not used
13	PUSHM 1
14	STO
15	JSS
16	STO
17	SKEQ
18	not used
19	not used
1A	not used
1B	not used
1C	not used
1D	not used
1E	JSA0*
1F	STO

1- must be added in ALU test to
to complete coverage

TABLE 7I

3) ALU TEST

Tables 71 thru 75 indicate the coverage of the microinstruction fields accomplished thus far by tests (1) and (2). Also listed in these tables are the instructions that must be added to complete the coverage. The following test is used to exercise these instructions:

- 1) The PUSHF and PUSHM instructions are used to store the contents of the accumulators and flags prior to the cyclic sum check of the scratch pad test. This portion of the ALU test runs concurrently with the Scratch Pad and Memory Address Processor tests.
- 2) The POPM instruction is executed at the completion of the Memory Address Processor test to restore the accumulators to their values at the execution of the first cyclic sum check.
- 3) The cyclic sum check is repeated and the result is compared to the first value of the test value table.
- 4) The CONT instruction is used to toggle the OV, F1, and F2 flags.
- 5) The SROV, SSF1, and SSF2 instructions are then used to test the setting of these flags.
- 6) A test computation is performed using the remaining instructions indicated on these tables. Since F1 is reset these instructions are tested in the saturated mode.
- 7) The POPF instruction is used to restore the flags.
- 8) The setting of the flags is tested using the SSF2, SRF1, and SROV instructions.
- 9) The cyclic check sum test is used to check the results of the test computation.

COVERAGE OF ALU CONTROL FIELDS

FIELD	SOURCE	FUNCTION	DESTINATION
	U12, U11, U10	U15, U14, U13	U18, U17, U16
0	CLA	CLA	CLA
1	TRA	STO	DMPY 1
2	CLA	JSS	not used
3	IAR	EXOR 1	RLS
4	CLA	CLA	CONT
5	POPM	not used	DIV 1
6	STO	CONT	CLA
7	CONT	LCM	PUSHF

i- instructions must be added to ALU test for coverage

TABLE 72

COVERAGE OF SCRATCH PAD ADDRESSING CONTROL FIELD

FIELD	SP POINTER		SP MULTIPLEXER	
	U1, U2, U3		U4, U5, U6	
0 *	dc		CONT	
1 *	dc		CLA	
2 *	dc		LOAD	
3	STO		STO	
4 *	dc		DIV	i
5 *	dc		DMPY	i
6 *	dc		not used	
7	STO		STO	
8	not used		not used	
F	not used		not used	
13	not used		not used	
17	not used		not used	
1B	not used		not used	
1F	not used		not used	
23	not used		not used	
27	PUSHF		PUSHF	i
2D	not used		not used	
2F	not used		not used	
33	PUSHF		PUSHE	i
37	not used		not used	
3B	not used		not used	i- instruction must be added to ALU test for coverage dc-don't care
3F	RPS		RPS	

TABLE 73

COVERAGE OF CONDITION SELECT FIELD

FIELD	INSTRUCTION
U36, U37, U38, U39	
0	CONT
1	STO
2	DMPY
3	ADDR
4	MPY i
5	STO
6	ADDR
7	not used
8	not used
9	DIV
A	SKEQ
B	CMP
C	DMPY
D	PUSHF
E	used for I/O instructions
F	used for test panel interface

TABLE 74

COVERAGE OF STATUS FLAG LOGIC FIELD

FIELD	INSTRUCTION
U31,U32,U33,U34,U35	
0	CLA
1	PUSHP
2	STC
3	CONT
4	unused
5	
6	unused
7	unused
8	DIV
9	
A	
B	
C	DMPY
D	
E	
F	DIV
10	ADDR
11	DIV
12	
13	SUB
14	
15	CONT
16	unused
17	unused
18	DIV
19	
1A	
1B	
1C	
1D	
1E	unused
1F	

TABLE 75

COVERAGE OF SHIFT INMUX FIELD

FIELD	INSTRUCTION
U7,U8,U17	
0	CLA
1	CLA
2	DMPY
3	RLL t
4	IAR sat
5	DIV
6	not used
7	not used

TABLE 76

ORDER OF INSTRUCTION EXECUTION

INSTRUCTION

TEST

CONT	***	CLA ineffective because
CLA (IR)	*	emulator sets initializes accumulator to 0
STO*	*	
LOAD rel to P	*	
LOAD rel to A0	*	
TRA	*	SCRATCH PAD TEST
LCM	*	
ADDR	*	
IAR	*	
SKEQ	*	
PUSHF	*	
PUSHM	*	
JSS	*	
RLS	*	
RPS	*	
CMP rel to A0	*	
LOAD*	***	
SUB rel to A0	***	
JSA1 rel to A1	*	MEMORY ADDRESS PROCESSOR TEST
JSA0* rel to P	*	
JU rel to A1	*	
POPM	***	
SROV	***	
SSF1	*	
SSF2	*	
DMPY	*	
DIV sat	*	ALU TEST
ADDR sat	*	
IAR sat	*	
ACM	*	
DIV	*	
EXOR	*	
IR	*	
MPY	*	
POPF	*	
SSF2	*	
SRF1	*	
SROV	***	

TABLE 77 (continued on next page)

SRLA
STM*
JMAO*
RET
LDM*
SUB*
DADDR
SLSA
RLL
SLLL
SLLA
SRLI
SRSL
SRSA
PUSHM
PADDM
DACM

*
*
*
*
*
*
*
*
*
*
*
*
*
*
*

MICROPROPRAM TEST

TABLE 77

4) MICROPROGRAM PROM TEST

The PROMs used prom consists of row and column addressing logic besides the actual memory cells. The addressing logic is tested by simply generating enough addresses to cover each of the row and column addresses. However, it is difficult to establish the percentage of the chip is used for addressing. Therefore, this test concentrates on bit cell coverage and the addressing coverage is expected to fall out as a result. In this test it is difficult to get a high coverage. Nevertheless, the rest of the CPU is virtually completely covered. Hence, it is assumed that the 5 % of the portion of the CPU uncovered by this test will eventually be the memory bit cells. Instructions are added as the emulator determines coverage. The expected coverage of the microprogram memory by the previous tests was determined by examining the microprogram exercised by the instructions already executed (listed in table 77). The expected coverage was found to be 45.28%. Table 78 lists instructions added to increase the micromemory coverage in the order of their effectiveness. It should be noted that the size of the total microprogram (not including test panel or I/O) is 382 instructions.

ADDITIONAL MICROMEMORY COVERAGE

INSTRUCTION	INSTRUCTIONS COVERED	ACCUMULATIVE ADDITIONAL COVERAGE
JMAO	17	17
LDM	15	32
STM	15	47
ADD sat	14	61
DIV additional cond	14	75
SLSA sat	12	87
SUB sat	11	98
ADD*	9	107
DACM sat	9	116
DADDR sat	9	125
DSUBR	9	134
DADDR	4	138
RET	4	142

TABLE 78

4) MICROPROGRAM PROM TEST

The PROMs used prom consists of row and column addressing logic besides the actual memory cells. The addressing logic is tested by simply generating enough addresses to cover each of the row and column addresses. However, it is difficult to establish the percentage of the chip is used for addressing. Therefore, this test concentrates on bit cell coverage and the addressing coverage is expected to fall out as a result. In this test it is difficult to get a high coverage. Nevertheless, the rest of the CPU is virtually completely covered. Hence, it is assumed that the 5 % of the portion of the CPU uncovered by this test will eventually be the memory bit cells. Instructions are added as the emulator determines coverage. The expected coverage of the microprogram memory by the previous tests was determined by examining the microprogram exercised by the instructions already executed (listed in table 77). The expected coverage was found to be 45.28%. Table 78 lists instructions added to increase the micromemory coverage in the order of their effectiveness. It should be noted that the size of the total microprogram (not including test panel or I/O) is 382 instructions.

ADDITIONAL MICROMEMORY COVERAGE

INSTRUCTION	INSTRUCTIONS COVERED	ACCUMULATIVE ADDITIONAL COVERAGE
JMAO	17	17
LDM	15	32
STM	15	47
ADD sat	14	61
DIV additional cond	14	75
SLSA sat	12	87
SUB sat	11	98
ADD*	9	107
DACM sat	9	116
DADDR sat	9	125
DSUBR	9	134
DADDR	4	138
RET	4	142

TABLE 78

6 15 19A1

* CYCLIC RAM TEST FOR CPUT
SOURCE STATEMENTPROGRAM CPUT2 MREF STMT
LUC ORJ

5030

```

1  *****
2  ORG 5030H
3  *****
4
5
6
7  MODULE
8
9  UNIT
10
11
12
13  -- CPU SELF TEST (CPUTM)
14
15  -- CPU SELF TEST (CPUT)
16
17
18  DESIGNER
19  -- P. CURRAN
20
21  PROGRAMMER
22  -- P. CURRAN
23
24  LAST REVISED
25  -- A JUN 81 (02000) RCA
26
27  FUNCTION:
28
29  CPUT? DOES CYCLIC RAM TESTING FOR THE CPU SELF TEST.
30
31  PARAMETERS:
32
33  INPUT:
34  NONE
35
36  OUTPUT:
37  NONE
38
39  UNITS INVOKED:
40  NONE
41
42  DATA BASE MACROS INVOKED:
43
44  DBREG -- REGISTER EQUATE
45
46  DATA REFERENCED:
47  NONE
48
49  CYCLIC SUM TEST TIMING ESTIMATES:
50
51  NOMINAL TIME:
52  MAXIMUM TIME:
53
54  ROM USAGE:
55
56  CYCLIC SUM TEST CODE & LINKS -- 30 WORDS
57
58  RAM USAGE:
59  NONE
60
61  COMMENTS:
62  NONE
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

6 15 1981

* CYCLIC RAM TEST FOR CPUT

PROGRAM CPUT2

LOC ORJ

MRFF

STMT

SOURCE STATEMENT

57 *
58 *
59 * DATA RAMP MACRO CALLS
60 *

0000 EQU 0
0001 EQU 1
0002 EQU 2
0003 EQU 3
0004 EQU 4
0005 EQU 5
0006 EQU 6
0007 EQU 7
0008 EQU 8
0009 EQU 9
000A EQU 10
000H EQU 11
000C EQU 12
000D EQU 13
000E EQU 14
000F EQU 15

77 * ENTRY POINT

78 *

79 *

80 CPUT2 ENTRY CPUT2

81 *

82 SA EQU *

83 *

84 *

85 *

86 *

87 *

5030

STARTING ADDRESS

* CYCLIC RAM TEST

5030 0301 RLS A0,1
5031 0801 ADDR A0,A1
5032 0301 RLS A0,1
5033 0802 ADDR A0,A2
5034 0301 RLS A0,1
5035 0803 ADDR A0,A3
5036 0301 RLS A0,1
5037 0804 ADDR A0,A4
5038 0301 RLS A0,1
5039 0805 ADDR A0,A5
503A 0301 RLS A0,1
503B 0806 ADDR A0,A6
503C 0301 RLS A0,1
503D 0807 ADDR A0,A7
503E 0301 RLS A0,1
503F 0808 ADDR A0,A8
5040 0301 RLS A0,1
5041 0809 ADDR A0,A9
5042 0301 RLS A0,1
5043 080A ADDR A0,A10
5044 0301 RLS A0,1
5045 080H ADDR A0,A11
5046 0301 RLS A0,1
5047 080C ADDR A0,A12
5048 0301 RLS A0,1

6 15 1981

* CYCLIC RAM TEST FOR CP11T

PROGRAM CP11T2	LOC ORJ	MREF	STMT	SOURCE STATEMENT
5049 0A0D			113	ADDR A0,A13
504A 0301			114	RLS A0,1
504B 0A0E			115	ADDR A0,A14
			116 *	
			117 *	RETURN TO CALLING UNIT
			118 *	
			119 *	RPS 0
504C FF001200			120 *	
			121 *	
			122 *	
			123 *	NO LINKS OR LOCAL DATA
			124 *	
	001E		125 US	EQU --SA
			126 *	
504E			127	END

UNIT SIZE

ENTRY SYMBOL TABLE
CPU12

EXTRN SYMBOL TABLE

PROGRAM CPU12 HAS 0 ERRORS

CROSS REFERENCE TABLE														
SYMBOL	VALUE	DEFINITION	REFERENCES											
A0	0000	61	88 89 10A 109 A9	90 91 92 93 94 95	96 97 98 99 100 101 102 103 104 105 106 107									
A1	0001	62	A9											
A10	000A	71	107											
A11	000B	72	109											
A12	000C	73	111											
A13	000D	74	113											
A14	000E	75	115											
A15	000F	76												
A2	0002	63	91											
A3	0003	64	93											
A4	0004	65	95											
A5	0005	66	97											
A6	0006	67	99											
A7	0007	68	101											
A8	0008	69	103											
A9	0009	70	105											
CPUT2	5030	A0	A0											
SA	5030	A2	125											
S	001E	125												

6 15 1981

```

PROGRAM CPU1
LOC OBJ
504E 0000
4F00

* CPU TEST
SOURCE STATEMENT
-----

```

```

      ABS
      ORG 4F00H
      EXTRN CPU2?
*****
      MODULE
      UNIT
      -- CPU SELF TEST (CPU1M)
      -- CPU SELF TEST (CPU2)

```

```

      DESIGNER
      PROGRAMMER
      LAST REVISED
      -- 8 JUN 81 1020001 RCH
      -- 8 JUN 81

```

```

      FUNCTION:

```

THE CPU TEST VERIFIES THE INTEGRITY OF THE PROCESSOR
PRIOR TO FLIGHT WHEN INVOKED BY RTT AND DURING FLIGHT WHEN INVOKED
BY SELF TEST. IT CONSISTS OF FOUR FUNCTIONAL AREAS AS FOLLOWS:

1. ACCUMULATOR SCATCH PAD TEST
2. MEMORY ADDRESS PROCESSOR TEST
3. ALU TEST
4. MICROPROGRAM MEMORY TEST

```

      PARAMETERS:

```

```

      INPUT:

```

```

      NONE

```

```

      OUTPUT:

```

```

      NONE

```

```

      UNITS INVOKED:

```

```

      CPU2 -- 'CYCLIC SUM TEST IN SYSMN.CPUTM'

```

```

      DATA BASE MACROS INVOKED:

```

```

      DMWFC -- REGISTER EQUATE

```

```

      DMIST -- TEST STATUS TABLE

```

```

      DATA REFERENCED:

```

```

      TEST 'TEST STATUS TABLE IN FSST'

```

```

      TSCPU 'CPU SCRATCHPAD IN CPU'

```

```

      CPU2 'CPU TEST RESULT' - /SET/

```

```

      TSSTS 'SELF TEST STACK AREA IN CPU'

```

```

      SAVE 'SAVE AREA USED BY CPU TEST' - /SET,USE/

```

```

      TSHT 'DATA LINK TEST RECEIVE BUFFER IN SD11' - /SET,USE/

```

6 15 1981

```

PROGRAM CPUT      * CPU TEST
LOC  OHJ          SOURCE STATEMENT
-----
57 * CYCLIC SUM TEST TIMING ESTIMATES:
58 *
59 *
60 *   NOMINAL TIME: 500 US
61 *   MAXIMUM TIME: 500 US
62 *
63 *   ROM USAGE:
64 *
65 *   CPU TEST CONF & LINKS -- 277 WORDS
66 *   CPU TEST LOCAL DATA -- 23 WORDS
67 *
68 *   RAM USAGE:
69 *   NONE
70 *
71 *   COMMENTS:
72 *   NONE
73 *
74 *****
75 *
76 *   DATA BASE MACRO CALLS
77 *   DBREG
78 *
79 A1 EQU 1
80 A0 EQU 0
81 A2 EQU 2
82 A3 EQU 3
83 A4 EQU 4
84 A5 EQU 5
85 A6 EQU 6
86 A7 EQU 7
87 A8 EQU 8
88 A9 EQU 9
89 A10 EQU 10
90 A11 EQU 11
91 A12 EQU 12
92 A13 EQU 13
93 A14 EQU 14
94 A15 EQU 15
95 *   DATST
96 TSCPIH EQU 2050H
97 TSXPT EQU 0
98 TSXPT+1
99 TSXPE EQU TSXPT+1
100 TSXTP EQU TSXPE+1
101 TSXTP+1
102 TSADC EQU TSXTP+1
103 TSAMD EQU TSADC+1
104 TSMDN EQU TSAMD+1
105 TSMDS EQU TSMDN+1
106 TSSTC EQU TSMDS+1
107 TSSTR EQU TSSTC+1
108 DGVAL EQU TSSTR+1
109 DLVAL EQU DGVAL+1
110 FLCID EQU DLVAL+1
111 FLWDI EQU FLCID+1
112 TSCWP EQU FLWDI+1

0001
0000
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F

2050
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
000D
000E
000F

2050H
0
TSXPT+1
TSXPE+1
TSXTP+1
TSXTP+1
TSADC+1
TSAMD+1
TSMDN+1
TSMDS+1
TSSTC+1
TSSTR+1
TSSTR+1
DGVAL+1
DLVAL+1
FLCID+1
FLWDI+1
TSCWP+1

CPU SCRATCH PAD ADDRESS.
INTERRUPT <XPT> INDICATOR WORD.
FRAMES TIL NO <S>OFT <P>ARITY <E>RROR COUNT.
CURRENT <D>IAGNOSTIC <T>EST <M>ODE.

```

113 TSPST EQU TSCW2+1
114 CRSLT EQU TSPST+1
115 TSSIS EQU 3C00H
116 SAVE EQU 0
117 TSBLT EQU 3000H
118 *
119 * EXTERNAL UNITS REFERENCED
120 *
121 *
122 * ENTRY POINT
123 *
124 * ENTRY CPUIT
125 *
126 SA EQU *
127 *
128 *
129 *
130 *
131 *
132 *
133 CPUIT PUSHM A0,A14
134 *
135 * CPU TEST
136 *

SELF TEST STACK AREA STARTING ADDRESS
STACK USED BY CPU SELF TEST
DATA LINK BUFFER LEFT

STARTING ADDRESS

* SAVE CALLING ROUTINES REGISTERS

SAVE STACK POINTER

CLEAR TEST RESULT
START ACCUMULATOR SCRATCH PAD TEST
(1-1) FILL ACCUMULATORS WITH TEST PATTERN

(1-2),(1-3),(1-4)

4F02 0465 CONT 1S,OR,ER,2R
4F03 000F TRA A0,A15
4F04 E458 STO* A0,LKFRP
4F05 7D0A9554 LDM* A0,10,LKCPT
4F07 7E0A9556 STM* A0,10,LKMP2
4F09 0900 CLA A0
4F0A E44E STO* A0,LKCRS
4F0B 544F LOAD A0,1KTAB
4F0C 5902 LOAD A1,2,A0
4F0D 0021 TRA A2,A1
4F0E 5903 LOAD A1,3,A0
4F0F 0031 TRA A3,A1
4F10 5904 LOAD A1,4,A0
4F11 0041 TRA A4,A1
4F12 5905 LOAD A1,5,A0
4F13 0051 TRA A5,A1
4F14 5906 LOAD A1,6,A0
4F15 0061 TRA A6,A1
4F16 5907 LOAD A1,7,A0
4F17 0071 TRA A7,A1
4F18 5908 LOAD A1,8,A0
4F19 0081 TRA A8,A1
4F1A 5909 LOAD A1,9,A0
4F1B 0091 TRA A9,A1
4F1C 590A LOAD A1,10,A0
4F1D 00A1 TRA A10,A1
4F1E 590H LOAD A1,11,A0
4F1F 00H1 TRA A11,A1
4F20 590C LOAD A1,12,A0
4F21 00C1 TRA A12,A1
4F22 590D LOAD A1,13,A0
4F23 00D1 TRA A13,A1

PROGRAM CPUIT	LOC	ORJ	MRIF	STMT	* CPU TEST SOURCE STATEMENT	
4F24	590F			169	LOAD A1,14,A0	
4F25	00E1			170	TRA A14,A1	
4F26	590F			171	LOAD A1,15,A0	
4F27	00E1			172	TRA A15,A1	
4F28	02FF			173	LCM A15,A15	
4F29	081F			174	ADDR A1,A15	
4F2A	8F11			175	IAR A1,1	
4F2B	8A11			176	SKEQ A1,CP000	
4F2C	942A		4F2D	177	JU* LKNDX	
4F2D	02FF		4F56	178	CP000 LCM A15,A15	
4F2E	5901			179	LOAD A1,1,A0	
4F2F	5A00			180	LOAD A0,0,A0	
4F30	FF0046FF			181	PUSHF	
4F32	7F0E420F			182	PUSHM A0,A14	
4F34	FF009423			183	JSS* LKCYL	
4F36	5524		4F57	184	LOAD A1,LKTAR	
4F37	4C10		4F5A	185	CMP A0,16,A1	
4F38	1403		4F3B	186	JU CP005	
4F39	941D		4F56	187	JU* LKNDX	
4F3A	941C		4F56	188	JU* LKNDX	
4F3B	D41D		4F58	189	CP005 LOAD* A0,LKCRS	
4F3C	8F01			190	IAR A0,1	
4F3D	E41A		4F58	191	STO* A0,LKCRS	
4F3E	02EE			192	LCM A14,A14	
4F3F	02DD			193	LCM A13,A13	
4F40	02CC			194	LCM A12,A12	
4F41	028H			195	LCM A11,A11	
4F42	02AA			196	LCM A10,A10	
4F43	0299			197	LCM A9,A9	
4F44	0288			198	LCM A8,A8	
4F45	0277			199	LCM A7,A7	
4F46	0266			200	LCM A6,A6	
4F47	0255			201	LCM A5,A5	
4F48	0244			202	LCM A4,A4	
4F49	0233			203	LCM A3,A3	
4F4A	0222			204	LCM A2,A2	
4F4B	5C00			205	LOAD A0,0,A1	
4F4C	5D01			206	LOAD A1,1,A1	
4F4D	0200			207	LCM A0,A0	
4F4E	0211			208	LCM A1,A1	
4F4F	FF009408		4F57	209	JSS* LKCYL	
4F51	5509		4F5A	210	LOAD A1,LKTAR	
4F52	4C11			211	CMP A0,17,A1	
4F53	140H		4F5E	212	JU CP010	
4F54	9402		4F56	213	JU* LKNDX	
4F55	9401		4F56	214	JU* LKNDX	
				215	*****	
				216	*****	
				217	*****	
				218	* LINK SECTION	
				219	*	
4F56	4FFE			220	LKNDX LINK CP070	
4F57	5030			221	LKCYL LINK CPU12	
4F58	206H			222	LKCRS LINK TSCPU+CRSLT	
4F59	5022			223	LKOPT LINK LD005	
4F5A	500H			224	LKTAR LINK LDCPV	

297

6 15 1981

PROGRAM CPUIT	LUC	INJ	MRFF	STMT	* CPUIT	SOURCE STATEMENT
4F93 090F				281	IR	A0,A15
4F94 7F0E9571		5005		282	STM*	A0,A14,LK9AV
4F96 0911				283	CLA	A1
4F97 0922				284	CLA	A2
4F98 0933				285	CLA	A3
4F99 0944				286	CLA	A4
4F9A 0955				287	CLA	A5
4F9H 0966				288	CLA	A6
4F9C 0977				289	CLA	A7
4F9D 0988				290	CLA	A8
4F9F 0999				291	CLA	A9
4F9E 09AA				292	CLA	A10
4FA0 09BH				293	CLA	A11
4FA1 09CC				294	CLA	A12
4FA2 09DD				295	CLA	A13
4FA3 09EE				296	CLA	A14
4FA4 7F0D420F				297	PUSHM	A0,A13
4FA6 7F0D400F				298	PADDM	A0,A14
4FA8 AE02		4FAB		299	SKNF	A0,CP035
4FA9 7D0E955C		5005		300	LDM*	A0,A14,LK9AV
4FAH 09CE				301	MPY	A12,A14
4FAC 7E0E9559		5005		302	STM*	A0,A14,LK9AV
4FAE FF0094A9		4F57		303	J99*	LKCYL
4FR0 D557		5007		304	LOAD*	A1,LKTRL
4FB1 3C13				305	SUB	A0,19,A1
4FB2 RA01		4FB4		306	SKED	A0,CP040
4FB3 144R		4FFE		307	JU	CP070
4FB4 D556		500A		308	LOAD*	A1,LKRST
4FH5 RF11				309	IAR	A1,1
4FH6 E554		500A		310	STO*	A1,LKRST
4FH7 FF00450F				311	POPF	
4FH9 RRF2		4FRC		312	SSF2	CP045
4FHA RA61		4FRC		313	SRF1	CP045
4FHR RA41		4FRD		314	SROV	CP050
4FNC 1442		4FFE		315	JU	CP045
4FHD D54D		500A		316	LOAD*	A1,LKRST
4FHE RF11				317	IAR	A1,1
4FHF E54H		500A		318	STO*	A1,LKRST
4FC0 FF0020A6				319	DMPY	A10,A6
4FC2 FF002024				320	DMPY	A2,A4
4FC4 FF002068				321	DMPY	A6,AR
4FC6 FF0020AC				322	DMPY	AR,A12
4FC8 A943				323	SRLA	A4,3
4FC9 F1AC				324	DSURR	A10,A12
4FCA F0A2				325	DADDR	A10,A2
4FCR A43D		500A		326	ADD*	A0,LKCTV
4FCC 9702		4FCE		327	JMA0*	CP055
4FCD 1431		4FFE		328	JU	CP070
4FCF 4FCF				329	LINK	CP060
4FCF F004				330	RET	4
4FD0 142E		4FFE		331	JU	CP070
4FD1 0F40				332	DIV	A4,A0
4FD2 0F27				333	DIV	A2,A7
4FD3 FF009484		4F57		334	JSS*	LKCYL
4FD5 0532		5007		335	LOAD*	A1,LKTRL
4FD6 3C14				336	SUB	A0,P0,A1

MICROPROGRAM TEST

6 15 1981

PROGRAM CPU#	LOC	OBJ	MREF	STMT	CPU TEST	SOURCE STATEMENT
5008 FFFF				393	*	*****
500C D0FF				394	*	*****
500D EF1A				395	*	*****
500E A130				396	*	*****
500F A6E0					*	*****
5010 012A					*	*****
5011 A24C					*	*****
5012 F011					*	*****
5013 DF31					*	*****
5014 C9F9					*	*****
5015 108F					*	*****
5016 C5EE					*	*****
5017 A6A9					*	*****
5018 D3RR					*	*****
5019 1001					*	*****
501A 3C01					*	*****
501B D83C					*	*****
501C F4C5					*	*****
501D FA1A					*	*****
501E F1F0					*	*****
501F 35F2					*	*****
5020 48FB					*	*****
5021 0000					*	*****
				420	*	*****
				421	LD	EQU **SA-IIS1
				422	*	*****
				423	*	*****
				424	*	*****
5022 AF12				425	LD005	IAR A1,2
5023 9704				426	JMA0*	LD015
5024 AF21			5027	427	LD010	IAR A2,1
5025 FD05				428	RET	5
5026 FA1A				429	FIX	LQMTS
5027 3002				430	LD015	LINK LQCRX
5028 1403				431	LD020	JU LD025
5029 83CE			5028	432	SKLT	A12,LD020
502A 8F2F				433	IAR	A2,-1
502B 1CFF				434	LD025	JU -1,A1
				435	*	*****
				436	HS	EQU **SA-LD
				437	*	*****
				438	*	*****
				439	*	*****
				440	*	*****
				441	*	*****
				442	*	*****
				443	*	*****
				444	*	*****
				445	*	*****
				446	*	*****
				447	*	*****
				448	*	*****
				449	*	*****
				450	*	*****
				451	*	*****
				452	*	*****
				453	*	*****
				454	*	*****
				455	*	*****
				456	*	*****
				457	*	*****
				458	*	*****
				459	*	*****
				460	*	*****
				461	*	*****
				462	*	*****
				463	*	*****
				464	*	*****
				465	*	*****
				466	*	*****
				467	*	*****
				468	*	*****
				469	*	*****
				470	*	*****
				471	*	*****
				472	*	*****
				473	*	*****
				474	*	*****
				475	*	*****
				476	*	*****
				477	*	*****
				478	*	*****
				479	*	*****
				480	*	*****
				481	*	*****
				482	*	*****
				483	*	*****
				484	*	*****
				485	*	*****
				486	*	*****
				487	*	*****
				488	*	*****
				489	*	*****
				490	*	*****
				491	*	*****
				492	*	*****
				493	*	*****
				494	*	*****
				495	*	*****
				496	*	*****
				497	*	*****
				498	*	*****
				499	*	*****
				500	*	*****
				501	*	*****
				502	*	*****
				503	*	*****
				504	*	*****
				505	*	*****
				506	*	*****
				507	*	*****
				508	*	*****
				509	*	*****
				510	*	*****
				511	*	*****
				512	*	*****
				513	*	*****
				514	*	*****
				515	*	*****
				516	*	*****
				517	*	*****
				518	*	*****
				519	*	*****
				520	*	*****
				521	*	*****
				522	*	*****
				523	*	*****
				524	*	*****
				525	*	*****
				526	*	*****
				527	*	*****
				528	*	*****
				529	*	*****
				530	*	*****
				531	*	*****
				532	*	*****
				533	*	*****
				534	*	*****
				535	*	*****
				536	*	*****
				537	*	*****
				538	*	*****
				539	*	*****
				540	*	*****
				541	*	*****
				542	*	*****
				543	*	*****
				544	*	*****
				545	*	*****
				546	*	*****
				547	*	*****
				548	*	*****
				549	*	*****
				550	*	*****
				551	*	*****
				552	*	*****
				553	*	*****
				554	*	*****
				555	*	*****
				556	*	*****
				557	*	*****
				558	*	*****
				559	*	*****
				560	*	*****
				561	*	*****
				562	*	*****
				563	*	*****
				564	*	*****
				565	*	*****
				566	*	*****
				567	*	*****
				568	*	*****
				569	*	*****
				570	*	*****
				571	*	*****
				572	*	*****
				573	*	*****
				574	*	*****
				575	*	*****
				576	*	*****
				577	*	*****
				578	*	*****
				579	*	*****
				580	*	*****
				581	*	*****
				582	*	*****
				583	*	*****
				584	*	*****
				585	*	*****
				586	*	*****
				587	*	*****
				588	*	*****
				589	*	*****
				590	*	*****
				591	*	*****
				592	*	*****
				593	*	*****
				594	*	*****
				595	*	*****
				596	*	*****
				597	*	*****
				598	*	*****
				599	*	*****
				600	*	*****
				601	*	*****
				602	*	*****
				603	*	*****
				604	*	*****
				605	*	*****
				606	*	*****
				607	*	*****
				608	*	*****
				609	*	*****
				610	*	*****
				611	*	*****
				612	*	*****
				613	*	*****
				614	*	*****
				615	*	*****
				616	*	*****
				617	*	*****
				618	*	*****
				619	*	*****
				620	*	*****
				621	*	*****
				622	*	*****
				623	*	*****
				624	*	*****
				625	*	*****
				626	*	*****
				627	*	*****
				628	*	*****
				629	*	*****
				630	*	*****
				631	*	*****
				632	*	*****
				633	*	*****
				634	*	*****
				635	*	*****
				636	*	*****
				637	*	*****
				638	*	*****
				639	*	*****
				640	*	*****
				641	*	*****
				642	*	*****
				643	*	*****
				644	*	*****
				645	*	*****
				646	*	*****
				647	*	*****
				648	*	*****
				649	*	*****
				650	*	*****
				651	*	*****
				652	*	*****
				653	*	*****
				654	*	*****
				655	*	*****
				656	*	*****
				657	*	*****
				658	*	*****
				659	*	*****
				660	*	*****
				661	*	*****
				662	*	*****
				663	*	*****
				664	*	*****
				665	*	*****
				666	*	*****
				667	*	*****
				668	*	*****
				669	*	*****
				670	*	*****
				671	*	*****
				672	*	*****
				673	*	*****
				674	*	*****
				675	*	*****
				676	*	*****
				677	*	*****
				678	*	*****
				679	*	*****
				680	*	*****
				681	*	*****
				682	*	*****
				683	*	*****
				684	*	*****
				685	*	*****
				686	*	*****
				687	*	*****
				688	*	*****
				689	*	*****
				690	*	*****
				691	*	*****
				692	*	*****
				693	*	*****
				694		

[illegible]

ENTRY SYMBOL TABLE
CPUT

EXTRN SYMBOL TABLE
CPUT2

PROGRAM CPUT HAS 0 ERRORS

SYMBOL	VALUE	DEFINITION	REFERENCE	REFERENCE	REFERENCE
LKCTV	5008	387	326	345	346
LKCYC	5009	388	364		
LKCYL	4F57	221	183	209	250
LKFRP	4F5C	226	139		303 334
LKMP1	4F5B	225	234	243	
LKMP2	4F5D	227	141		
LKNIX	4F56	220	177	187	188
LKRST	500A	389	308	310	316
LKSAV	5005	384	282	300	302
LKTAH	4F5A	224	144	184	210
LKTHL	5007	386	304	335	365
LKCPX	3000	443	225	227	444
LKCPX	3002	444	430		
LKMTS	ER1A	446	415	429	
LOXX	FFFF	445	225	235	245
SA	4F00	126	391	421	436
SAVE	0000	116	226	384	385
TSADC	0004	101	102		412
TSAMD	0005	102	103		
TSHIR	0009	106	107		
TSHIT	0006	103	104		
TSHLT	3000	117	443		
TSHTM	0003	100	101		
TSCPU	2050	96	222	389	
TSCW2	0016	112	113		
TSDTM	0002	99	100		
TSMDS	0007	104	105		
TSMIR	000A	107	108		
TSPST	0017	113	114		
TSSPE	0001	9A	99		
TSSTC	000A	105	106		
TSSTS	3C00	115	226	384	385
TSXPT	0000	97	98		412
US	0115	436			
US1	010R	391	421		

GLOBAL SYMBOL TABLE
CPUT 4F00 CPUT2 5030

PROGRAMS HAVE 0 ERRORS

6 15 1981

* CYCLIC RAM TEST FOR CPU1

SOURCE STATEMENT

PROGRAM CPUT2
LOC ORJ
5030

```

1  *PS
2  ORG 5030H
3  *****
4
5
6
7  MODULE
8
9  UNIT
10
11
12
13  DESIGNER
14
15  PROGRAMMER
16
17  LAST REVISED
18  -- 8 JUN 81 (02000) RCR
19
20  FUNCTION:
21
22  CPUT2 DOES CYCLIC RAM TESTING FOR THE CPU SELF TEST.
23
24  PARAMETERS:
25
26  INPUT:
27  NONE
28
29  OUTPUT:
30  NONE
31
32  UNITS INVOKED:
33  NONE
34
35  DATA BASE MACROS INVOKED:
36
37  ORREG'-- REGISTER EQUATE
38
39  DATA REFERENCED:
40  NONE
41
42  CYCLIC SUM TEST TIMING ESTIMATES:
43
44  NOMINAL TIME:
45  MAXIMUM TIME:
46
47  ROM USAGE:
48
49  CYCLIC SUM TEST CODE & LINKS -- 30 WORDS
50
51  RAM USAGE:
52  NONE
53
54  COMMENTS:
55  NONE
56

```

6 15 1981

* CYCLIC RAM TEST FOR CPUT

PROGRAM CPUT2
LOC (HJ) MREF SYMT

LOC	(HJ)	MREF	SYMT	DATA	MACRO CALLS	ENTRY POINT	STARTING ADDRESS
57							
58							
59							
60							
61							
62							
63							
64							
65							
66							
67							
68							
69							
70							
71							
72							
73							
74							
75							
76							
77							
78							
79							
80							
81							
82							
83							
84							
85							
86							
87							
88							
89							
90							
91							
92							
93							
94							
95							
96							
97							
98							
99							
100							
101							
102							
103							
104							
105							
106							
107							
108							
109							
110							
111							
112							

6 15 1981

* CYCLIC RAM TEST FOR CPUT

LOC	ORJ	MREF	STMT
5049	ORND		113
504A	0101		114
504B	0A0E		115
504C	FF001200		116
			117
			118
			119
			120
			121
			122
			123
			124
			125
			126
			127

SOURCE STATEMENT

```

      ADDR A0,A13
      RLS A0,1
      ADDR A0,A14
      RETURN TO CALLING UNIT
      RPS 0
      *****
      NO LINKS OR LOCAL DATA
      ECU *-SA
      END
  
```

UNIT SIZE

ENTRY SYMBOL TABLE
CPUT2

EXTRN SYMBOL TABLE

PROGRAM CPUT2 HAS 0 ERRORS

CROSS REFERENCE TABLE

PAGE 1

SYMBOL	VALUE	DEFINITION	REFERENCES
A0	0000	61	88 89 108 109 110 111 112 113 114 115
A1	0001	62	89
A10	000A	71	107
A11	000B	72	109
A12	000C	73	111
A13	000D	74	113
A14	000E	75	115
A15	000F	76	
A2	0002	63	91
A3	0003	64	93
A4	0004	65	95
A5	0005	66	97
A6	0006	67	99
A7	0007	68	101
A8	0008	69	103
A9	0009	70	105
CPUT2	5030	80	
SA	5030	A2	125
115	001E	125	

DO NOT READ THIS FOR (000)

DO NOT READ THIS

DO NOT READ THIS

DO NOT READ THIS

DO NOT READ THIS

DO NOT READ THIS

DO NOT READ THIS

DO NOT READ THIS

DO NOT READ THIS

DO NOT READ THIS

DO NOT READ THIS

DO NOT READ THIS

PROGRAM CPU
LOC 0HJ
504F 0000
4F00

* CPU TEST
SOURCE STATEMENT

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56

```

-----
ABS
ORG 4F00H
EXTRN CPUT2
*****
MODULE
UNIT
-- CPU SELF TEST (CPUTM)
-- CPU SELF TEST (CPUT)

DESIGNER -- P. CURRAN
PROGRAMMER -- P. CURRAN

LAST REVISED -- 8 JUN 81 (02000) RCB -- 8 JUN 81
FUNCTION:

THE CPU TEST VERIFIES THE INTEGRITY OF THE PROCESSOR
PRIOR TO FLIGHT WHEN INVOKED BY RTT AND DURING FLIGHT WHEN INVOKED
BY SELF TEST. IT CONSISTS OF FOUR FUNCTIONAL AREAS AS FOLLOWS:

1. ACCUMULATOR SCATCH PAD TEST
2. MEMORY ADDRESS PROCESSOR TEST
3. ALU TEST
4. MICROPROGRAM MEMORY TEST

PARAMETERS:
INPUT:
NONE
OUTPUT:
NONE
UNITS INVOKED:

CPUT2 -- 'CYCLIC SUM TEST IN SYSMN.CPUTM'

DATA BASE MACROS INVOKED:
DHREG -- REGISTER EQUATE
DHST -- TEST STATUS TABLE
DATA REFERENCED:

TST 'TEST STATUS TABLE IN FSST'
TSCPU 'CPU SCRATCHPAD IN CPU'
CHSLT 'CPU TEST RESULT' - /SET/
TSS19 'SELF TEST STACK AREA IN CPU'
SAVE 'SAVE AREA USED BY CPU TEST' - /SET,USE/
TSHLT 'DATA LINK TEST RECEIVE BUFFER IN SOLL' - /SET,USE/
    
```


6 15 1981

```

PROGRAM CPUT          * CPU TEST
LOC  (RJ)             SOURCE STATEMENT
-----
0017      TSPST EQU   TSCW2+1
0018      CRSLT EQU   TSPST+1
3000      TSSTS FOU   3000H      SELF TEST STACK AREA STARTING ADDRESS
0000      SAVE  EQU   0          STACK USED BY CPU SELF TEST
3000      TSBLT EQU   3000H      DATA LINK BUFFER LEFT

118      * EXTERNAL UNITS REFERENCED
119
120
121      * ENTRY POINT
122
123      * ENTRY CPUT
124
125      *
126      SA EQU *
127
128
129
130
131      * SAVE CALLING ROUTINES REGISTERS
132
133      CPUT PUSHM  A0,A14
134
135      * CPU TEST
136
137      CONT 19,OR,ER,2R
138      TRA A0,A15
139      STO* A0,LKFRP
140      LDM* A0,10,LKCPT
141      STM* A0,10,LKMP2
142      CLA A0
143      STO* A0,LKCRS
144      LOAD A0,LKTAB
145      LOAD A1,2,A0
146      TRA A2,A1
147      LOAD A1,3,A0
148      TRA A3,A1
149      LOAD A1,4,A0
150      TRA A4,A1
151      LOAD A1,5,A0
152      TRA A5,A1
153      LOAD A1,6,A0
154      TRA A6,A1
155      LOAD A1,7,A0
156      TRA A7,A1
157      LOAD A1,8,A0
158      TRA A8,A1
159      LOAD A1,9,A0
160      TRA A9,A1
161      LDC 1,10,A0
162      T 3,A1
163      L -1,11,A0
164      T 11,A1
165      L -1,12,A0
166      T 12,A1
167      L 1,13,A0
168      A13,A1

4F00 7F0E420F

4F02 0465      CONT 19,OR,ER,2R
4F03 000F      TRA A0,A15
4F04 E458      STO* A0,LKFRP
4F05 7D0A9554 4F59 LDM* A0,10,LKCPT
4F07 7E0A9556 4F5D STM* A0,10,LKMP2
4F09 0900      CLA A0
4F0A E44E      STO* A0,LKCRS
4F0B 544F      LOAD A0,LKTAB
4F0C 5902      LOAD A1,2,A0
4F0D 0021      TRA A2,A1
4F0E 5903      LOAD A1,3,A0
4F0F 0031      TRA A3,A1
4F10 5904      LOAD A1,4,A0
4F11 0041      TRA A4,A1
4F12 5905      LOAD A1,5,A0
4F13 0051      TRA A5,A1
4F14 5906      LOAD A1,6,A0
4F15 0061      TRA A6,A1
4F16 5907      LOAD A1,7,A0
4F17 0071      TRA A7,A1
4F18 5908      LOAD A1,8,A0
4F19 0081      TRA A8,A1
4F1A 5909      LOAD A1,9,A0
4F1B 0091      TRA A9,A1
4F1C 590A      LDC 1,10,A0
4F1D 00A1      T 3,A1
4F1E 590B      L -1,11,A0
4F1F 00H1      T 11,A1
4F20 590C      L -1,12,A0
4F21 00C1      T 12,A1
4F22 590D      L 1,13,A0
4F23 00D1      A13,A1

```

6 15 19A1

```

* CPU TEST
LOC  ORJ  MRFF  STMT
-----
4F24 590E          LOAD  A1,14,A0
4F25 00E1          TRA   A14,A1
4F26 590F          LOAD  A1,15,A0
4F27 00F1          TRA   A15,A1
4F28 02FF          LCM   A15,A15
4F29 081F          ADDR  A1,A15
4F2A AF11          IAR   A1,1
4F2B AA11          SKED  A1,CP000
4F2C 942A          JU*   LKNDX
4F2E 5901          LCM   A15,A15
4F2F 5A00          LOAD  A1,1,A0
4F30 FF0046FF      PUSHF A0,A14
4F32 7F0E420F      PUSHM A0,A14
4F34 FF009423      JSS*  LKCYL
4F36 5524          LOAD  A1,LKTAB
4F37 4C10          CMP   A0,16,A1
4F38 1403          JU*   CP005
4F39 941D          JU*   LKNDX
4F3A 941C          JU*   LKNDX
4F3B D41D          CP005 LOAD* A0,LKCRS
4F3C 8F01          IAR   A0,1
4F3D E418          STO*  A0,LKCRS
4F3E 02EE          LCM   A14,A14
4F3F 02DD          LCM   A13,A13
4F40 02CC          LCM   A12,A12
4F41 02RR          LCM   A11,A11
4F42 02AA          LCM   A10,A10
4F43 0299          LCM   A9,A9
4F44 0288          LCM   A8,A8
4F45 0277          LCM   A7,A7
4F46 0266          LCM   A6,A6
4F47 0255          LCM   A5,A5
4F48 0244          LCM   A4,A4
4F49 0233          LCM   A3,A3
4F4A 0222          LCM   A2,A2
4F4B 5C00          LOAD  A0,0,A1
4F4C 5D01          LOAD* A1,0,A1
4F4D 0200          LCM   A0,A0
4F4E 0211          LCM   A1,A1
4F4F FF009408      JSS*  LKCYL
4F51 5509          LOAD  A1,LKTAB
4F52 4C11          CMP   A0,17,A1
4F53 1408          JU   CP010
4F54 9402          JU*  LKNDX
4F55 9401          JU*  LKNDX

*****
* LINK SECTION
*****
215
216
217
218
219
220 LKNDX LINK CP070
221 LKCYL LINK CP072
222 LKCRS LINK TSCPU+CRSLT
223 LKCPY LINK LK005
224 LKTAH LINK LDCPV

```

6 15 1981

PROGRAM CPU	LOC	ORJ	MREF	SMT	* CPU TEST	SOURCE STATEMENT
4F58 3001	4F58	3001		225	LKMP1 LINK	LOCPE-LQXX
4F5C 3000	4F5C	3000		226	LKMP LINK	TSSTS+SAVE
4F5D 3000	4F5D	3000		227	LKMP2 LINK	LOCPE
				228		
				229		
				230		
4F5E D4FA	4F5E	D4FA		231	CP010 LOAD*	A0,LKCRS
4F5F 8F01	4F5F	8F01		232	IAR	A0,1
4F60 E4F8	4F60	E4F8		233	STO*	A0,LKCRS
4F61 50FA	4F61	50FA		234	LOAD	A0,LKMP1
4F62 5903	4F62	5903		235	LOAD	A1,LQXX+4,A0
4F63 59F7	4F63	59F7		236	LOAD	A0,LKTAB
4F64 3912	4F64	3912		237	SUB	A1,18,A0
4F65 8A11	4F65	8A11		238	SKEQ	A1,CP015
4F66 94F0	4F66	94F0		239	JU*	LKNDX
4F67 06F1	4F67	06F1		240	CP015 LOAD*	A2,LKCRS
4F68 8F21	4F68	8F21		241	IAR	A2,1
4F69 E6EF	4F69	E6EF		242	STO*	A2,LKCRS
4F6A 55F1	4F6A	55F1		243	LOAD	A1,LKMP1
4F6B 5795	4F6B	5795		244	LOAD	A3,CPUT
4F6C 1EFF	4F6C	1EFF		245	JSA1	LQXX,A1
4F6D 94E9	4F6D	94E9		246	JU*	LKNDX
4F6E 8F21	4F6E	8F21		247	IAR	A2,1
4F6F E6E9	4F6F	E6E9		248	STO*	A2,LKCRS
4F70 7F0E410F	4F70	7F0E410F		249	POPM	A0,A14
4F72 FF0094E5	4F72	FF0094E5		250	JSS*	LKCYL
4F74 55E6	4F74	55E6		251	LOAD	A1,LKTAB
4F75 3C10	4F75	3C10		252	SUB	A0,16,A1
4F76 8A01	4F76	8A01		253	SKEQ	A0,CP020
4F77 940F	4F77	940F		254	JU*	LKNDX
4F78 05E0	4F78	05E0		255	CP020 LOAD*	A1,LKCRS
4F79 8F11	4F79	8F11		256	IAR	A1,1
4F7A E50E	4F7A	E50E		257	STO*	A1,LKCRS
4F7B 0492	4F7B	0492		258	CONT	05,2S,1R
4F7C 8H42	4F7C	8H42		259	SROV	CP025
4F7D 8HE1	4F7D	8HE1		260	SSF1	CP025
4F7E 8HF1	4F7E	8HF1		261	SSF2	CP030
4F7F 94D7	4F7F	94D7		262	CP025 JU*	LKNDX
4F80 8143	4F80	8143		263	CP030 SLLA	A4,3
4F81 0F61	4F81	0F61		264	DIV	A6,A1
4F82 0F42	4F82	0F42		265	DIV	A4,A2
4F83 0A6C	4F83	0A6C		266	ADDR	A6,A12
4F84 8F72	4F84	8F72		267	IAR	A7,2
4F85 0177	4F85	0177		268	ACM	A7,A7
4F86 0420	4F86	0420		269	CONT	1S
4F87 0F28	4F87	0F28		270	DIV	A2,A8
4F88 0F21	4F88	0F21		271	DIV	A2,A1
4F89 FF0020AC	4F89	FF0020AC		272	DMPY	A10,A12
4F8B F334	4F8B	F334		273	EXOR	A3,A4
4F8C 0233	4F8C	0233		274	LCM	A3,A3
4F8D F334	4F8D	F334		275	EXOR	A3,A4
4F8F 02FF	4F8F	02FF		276	LCM	A15,A15
4F8F 09F0	4F8F	09F0		277	IR	A15,A0
4F90 090F	4F90	090F		278	IR	A0,A15
4F91 02FF	4F91	02FF		279	LCM	A15,A15
4F92 09F0	4F92	09F0		280	IR	A15,A0

MEMORY ADDRESS PROCESSOR TEST
(2-1),(2-2)

(2-3) COMPARE WITH TEST RESULT

INCREMENT CRSLT

(2-4) CLEAR T REGISTER
(2-5)INCREMENT CRSLT
ALU TEST
(3-2)INCREMENT CRSLT
(3-4) TEST FLAGS

COMPUTATIONS WITH SATURATED ARITHMETIC

COMPUTATIONS WITH UNSATURATED ARITHMETIC

SCRATCH PAD TEST OF R OUT

PROGRAM CPUT	LOC	ORJ	MREF	STMT	* CPU TEST	SOURCE STATEMENT
4F93 090F	281				IR	A0,A15
4F94 7E0E9571	5005				STM*	A0,A14,LKSAV
4F96 0911	282				CLA	A1
4F97 0922	283				CLA	A2
4F98 0933	284				CLA	A3
4F99 0944	285				CLA	A4
4F9A 0955	286				CLA	A5
4F9B 0966	287				CLA	A6
4F9C 0977	288				CLA	A7
4F9D 0988	289				CLA	A8
4F9E 0999	290				CLA	A9
4F9F 09AA	291				CLA	A10
4FA0 09BB	292				CLA	A11
4FA1 09CC	293				CLA	A12
4FA2 09DD	294				CLA	A13
4FA3 09EE	295				CLA	A14
4FA4 7F0D420F	296				PUSHM	A0,A13
4FA6 7F0D400F	297				PADDM	A0,A14
4FA8 8E02	298				SKNCL	A0,CP035
4FA9 7D0E955C	5005				LDML	A0,A14,LKSAV
4FAB 08CE	301				MPY	A12,A14
4FAC 7E0E9559	5005				STM*	A0,A14,LKSAV
4FAE FF009A19	4F57				LDML	A0,A14,LKSAV
4FB0 D557	5007				LDML	A0,A14,LKSAV
4FB1 3C13	305				STM*	A0,A14,LKSAV
4FB2 8A01	4FB4				SKNCL	A0,CP035
4FB3 1448	4FFE				LDML	A0,A14,LKSAV
4FB4 D556	500A				LDML	A0,A14,LKSAV
4FB5 0F11	500A				LDML	A0,A14,LKSAV
4FB6 5554	500A				LDML	A0,A14,LKSAV
4FB7 FF00A50F	500A				LDML	A0,A14,LKSAV
4FB8 8BF2	4F8C				SKNCL	A0,CP035
4FB9 A861	4F8C				SKNCL	A0,CP035
4FBA 0841	4F8D				SKNCL	A0,CP035
4FBC 1442	4FFE				LDML	A0,A14,LKSAV
4FBD D54D	500A				LDML	A0,A14,LKSAV
4FBE AF11	500A				LDML	A0,A14,LKSAV
4FBF E548	500A				LDML	A0,A14,LKSAV
4FC0 FF0020A6	4FC0				LDML	A0,A14,LKSAV
4FC2 FF002024	4FC2				LDML	A0,A14,LKSAV
4FC4 FF002068	4FC4				LDML	A0,A14,LKSAV
4FC6 FF00208C	4FC6				LDML	A0,A14,LKSAV
4FC8 8A23	4FC8				LDML	A0,A14,LKSAV
4FC9 F1AC	4FC9				LDML	A0,A14,LKSAV
4FCA F0A2	4FCA				LDML	A0,A14,LKSAV
4FCB A430	4FCB				LDML	A0,A14,LKSAV
4FCC 9702	4FCC				LDML	A0,A14,LKSAV
4FCD 1431	4FCD				LDML	A0,A14,LKSAV
4FCE 4FCF	4FCE				LDML	A0,A14,LKSAV
4FCF F004	4FCF				LDML	A0,A14,LKSAV
4FD0 142E	4FD0				LDML	A0,A14,LKSAV
4FD1 0F40	4FD1				LDML	A0,A14,LKSAV
4FD2 0F27	4FD2				LDML	A0,A14,LKSAV
4FD3 FF009484	4F57				LDML	A0,A14,LKSAV
4FD5 0532	5007				LDML	A0,A14,LKSAV
4FD6 3C14	4FD6				LDML	A0,A14,LKSAV

MICROPROGRAM TEST

6 15 1981

```

PROGRAM CPUT
LOC  ORJ  MREF  STMT  SOURCE STATEMENT
-----
4FD7 RA02 4FDA 337 SKED A0,CP065
4FDA 1426 4FFE 338 JU CP070
4FD9 1425 4FFE 339 JU CP070
4FDA 0530 500A 340 CP065 LOAD* A1,LKRST
4FDH AF11 341 IAR A1,1
4FDC F52E 500A 342 STO* A1,LKRST
4FDD 700E9528 5005 343 LDM* A0,A14,LKSAV
4FDF 0410 344 CONT 1H
4FE0 A728 5008 345 ADD* 3,LKCTV
4FE1 R627 5008 346 SUB* 2,LKCTV
4FE2 F056 347 DIV DADDR A5,A6
4FE3 8088 348 LSR SLSA A8,8
4FE4 078C 349 LEA HLL A8,12
4FE5 8087 350 SLSA A8,7
4FE6 0420 351 CONT 1S
4FE7 85C3 352 SLLL A12,3
4FE8 A1A2 353 SLLA A10,2
4FE9 8D23 354 SLLL A2,3
4FEA AC31 355 SR9L A3,1
4FEB 8A61 356 SRSA A6,1
4FEC 7F0E420F 357 PUSHM A0,A14
4FEE 7F0E400F 358 PADDH A0,15,A15
4FF0 FF002122 359 OACM A2,A2
4FF2 0F25 360 DIV A2,A5
4FF3 0FAC 361 DIV A10,A12
4FF4 0761 362 RLL A6,1
4FF5 0066 363 TRA A6,A6
4FF6 FF009A13 5009 364 JSS* LKCYC
4FF8 D50F 5007 365 LOAD* A1,LKTRL
4FF9 3C15 366 SUB A0,21,A1
4FFA HE03 4FFE 367 SKNE A0,CP070
4FFB D50F 500A 368 LOAD* A1,LKRST
4FFC 8F11 369 IAR A1,1
4FFD E50D 500A 370 STO* A1,LKRST
4FFE D408 5006 371 CP070 LOAD* A0,LK2FP
4FFF 00F0 372 TRA A15,A0
5000 0428 373 CONT ES,1S
374 *
375 * RESTORE CALLING ROUTINES REGISTERS AND RETURN
376 *
377 * POPM A0,A14
378 * MPS 0
379 *
380 *****
381 * LINK SECTION
382 *
383 *
384 LKSAV LINK TSSTS+SAVE+17
385 LK2FP LINK TSSTS+SAVE
386 LKTHL LINK LKTAH
387 LKCTV LINK LDCPV+12
388 LKCYC LINK CPINT?
389 LKRST LINK TSCPU+CRSLT
390 *
391 HSI EQU -SA
392 *

5001 7F0E410F
5003 FF001200

5005 3C11
5006 3C00
5007 4F5A
5008 5017
5009 5030
500A 2068
010B

```

SATURATED ARITHMETIC

(3-10) TEST OF MUX WITH Q=0FFFFFH

RESTORE STACK POINTER

END OF SELF TEST

ENTRY SYMBOL TABLE
CPUT

EXTRN SYMBOL TABLE
CPUT2

PROGRAM CPUT HAS 0 ERRORS

DELETED BY 1000 10

9

100
100
100
100

SYMBOL	VALUE	DEFINITION	REFERENCES
LKCTV	5008	3A7	326 345 346
LKCYC	5009	3A8	364
LKCYL	4F57	221	183 209 250 303 334
LKFRP	4F5C	226	139
LKMP1	4F5B	225	214 243
LKMP2	4F5D	227	141
LKNIX	4F56	220	177 187 188 213 214 239 246 254 262
LKRST	500A	3A9	308 310 316 340 342 368 370
LKSAV	5005	3A4	282 300 302 343
LKTAH	4F5A	224	144 184 210 236 251 386
LKTHL	5007	3A6	304 335 365
LQCPH	3000	443	225 227 344 391 231 240 242 248 255 257
LQCPX	3002	444	430
LQMTS	ER1A	446	415 429
LQXX	FFFF	445	225 235 245
SA	4F00	126	391 421 436
SAVE	0000	116	226 384 385 412
TSADC	0004	101	102
TSAMD	0005	102	103
TSHIR	0009	106	107
TSHIT	0006	103	104
TSHLT	3000	117	443
TSHTM	0003	100	101
TSCPU	2050	96	222 389
TSCW2	0016	112	113
TSOTM	0002	99	100
TSMDS	0007	104	105
TSMIR	000A	107	108
TSPST	0017	113	114
TSSPE	0001	9A	99
TSSIC	0008	105	106
TSSIS	3C00	115	226 384 385 412
TSXPT	0000	97	98
US	0115	436	
US1	0108	391	421

PROGRAMS HAVE 0 ERRORS

—

— 1 —

1000

10

100

3

131

2

117

441

13

551

10

33-1-1

A

41

—

3.

4

Report Documentation Page

1. Report No. NASA CR-181642		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle Description of the BDX 930 Processor at the Gate Logic Level				5. Report Date April 1982	
				6. Performing Organization Code	
7. Author(s) F. Swern and J. McGough				8. Performing Organization Report No.	
				10. Work Unit No. 505-66-21-03	
9. Performing Organization Name and Address The Bendix Corporation Flight Systems Division Teterboro, NJ 07608				11. Contract or Grant No. NAS1-16807	
				13. Type of Report and Period Covered Contractor Report	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225				14. Sponsoring Agency Code	
15. Supplementary Notes Technical Monitor: Gerard E. Migneault Langley Research Center					
16. Abstract This document provides a gate-level description of the Bendix BDX-930 computer. The description is provided in sufficient detail, with signal names and other information so that a gate-level emulation of BDX-930 operation can be performed. The report consists of logic diagrams, logic equations and microcode listings.					
17. Key Words (Suggested by Author(s)) Gate Logic Description				18. Distribution Statement Unclassified-Unlimited Subject Category 62	
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of pages 324	
				22. Price	